
LYRTECH
LYRTECH SIGNAL PROCESSING


恒润科技
HIRAIN TECHNOLOGIES

基于模型的设计基础





课程概要

- Simulink建模仿真
- 面向DSP的自动代码生成
- 面向FPGA的自动代码生成



概述

- Simulink基础
- 模块的操作与参数的设置
- 仿真选项的设置和仿真机理
- 创建子系统与封装子系统
- Simulink 与MATLAB 工作区的接口
- 通过命令行仿真
- 模型浏览器Model Explore



Simulink 基础

- 用一些块组成的图形界面代表系统
- 使用一计算引擎，通过时间步长步进系统
  - 信号传递
  - 块的输出计算
- 使用 MATLAB 工作区参数
- 与 MATLAB 工作区有接口，可交换系统的输入、输出数据



启动 Simulink

MATLAB
File Edit Debug Desktop Window Help
c:\class\work
Shortcuts How to Add
Current Directory - c:\class\work
All Files
private
fun2.m
myhandles.mat
Current Directory Workspace
Command History
-- 3/17/04 1:35 PM --
clc
simulink
Command Window
```
>> simulink
>>
```
Start

Simulink Library Browser
File Edit View Help
Commonly Used Blocks: simulink/Commonly Used Blocks

- Simulink
  - Commonly Used Blocks
  - Continuous
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
  - User-Defined Functions
  - Additional Math & Discrete
- Aerospace Blockset
- CDMA Reference Blockset
- Communications Blockset
- Control System Toolbox
- Dials & Gauges Blockset
- Embedded Target for Infineon C166
- Embedded Target for Motorola® HC1
- Embedded Target for Motorola® MPC
- Embedded Target for OSEK/VDX
- Embedded Target for TI C6000 DSP

Commonly Used Blocks
Continuous
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks

Ready



Simulink 库浏览器

Simulink Library Browser

File Edit View Help

Bus Creator: This block creates a bus signal from its inputs.

Simulink
- Commonly Used Blocks
- Continuous
- Discontinuities
- Discrete
- Logic and Bit Operations
- Lookup Tables
- Math Operations
- Model Verification
- Model-Wide Utilities
- Ports & Subsystems
- Signal Attributes
- Signal Routing
- Sinks
- Sources
- User-Defined Functions
- Additional Math & Discrete

Aerospace Blockset

CDMA Reference Blockset

Bus Creator

Bus Selector

1 Constant

Convert Data Type Conversion

Demux

K Ts / z-1 Discrete-Time Integrator

1 Gain

Ground

1 In1

Ready

模块查询

模块说明

库

子库

库模块



创建新的模型框图

新建模型图标

Simulink 工具栏

Simulink Library Browser
File Edit View Help
New
Model Ctrl+N
Library
Open... Ctrl+O
Close
Preferences...

Simulink
Commonly Used Blocks
Continuous
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
CDMA Reference Blockset
Communications Blockset
Control System Toolbox
Dials & Gauges Blockset
Create a new Simulink model

untitled
File Edit View Simulation Format Tools Help
10.0
Normal
Go To Parent
Toolbar
Status Bar
Model Browser Options
Block Data Tips Options
System Requirements
Library Browser
Model Explorer
Zoom In
Zoom Out
Fit System To View
Normal (100%)
Remove Highlighting
Highlight ...
Show or hide the toolbar
100%
ode45

状态栏



收集和连接所需各块

用拖放操作收集模块

使用鼠标左键连接模块

Simulink Library Browser
File Edit View Help
Find
Derivative: Numerical derivative: du/dt.
Simulink
Continuous
Discontinuities
Discrete
Look-Up Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Aerospace Blockset
CDMA Reference Blockset
Communications Blockset

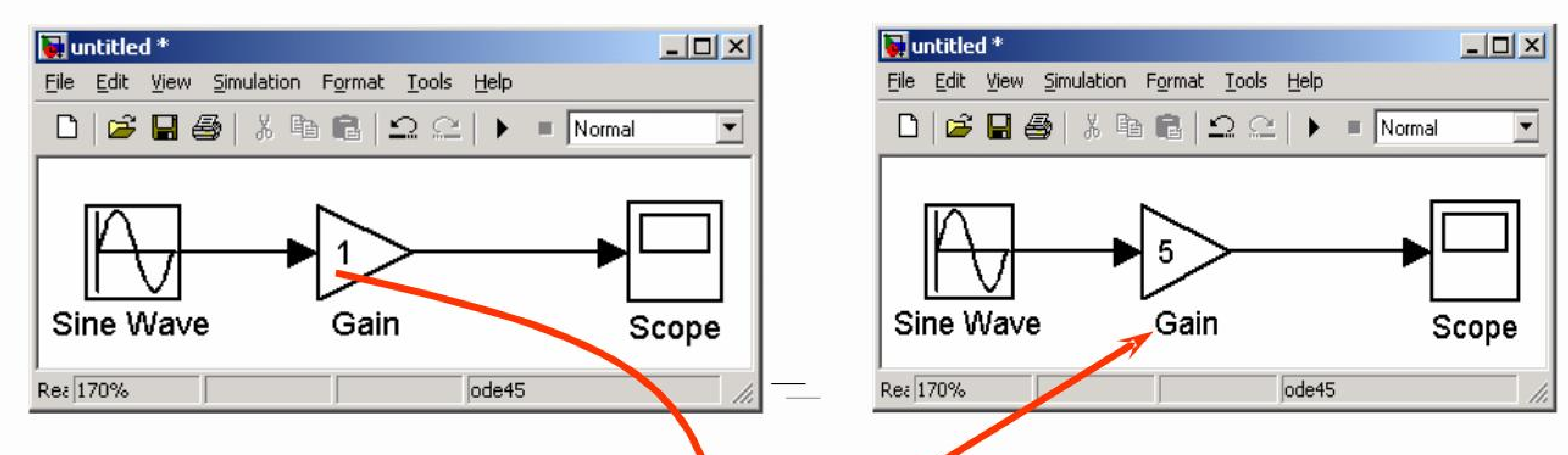
du/dt Derivative
Integrator
State-Space
Transfer Fcn
Transport Delay
Variable Transport Delay
Zero-Pole

untitled *
File Edit View Simulation Format Tools Help
Normal
Sine Wave
Gain
Scope
Ready 170%

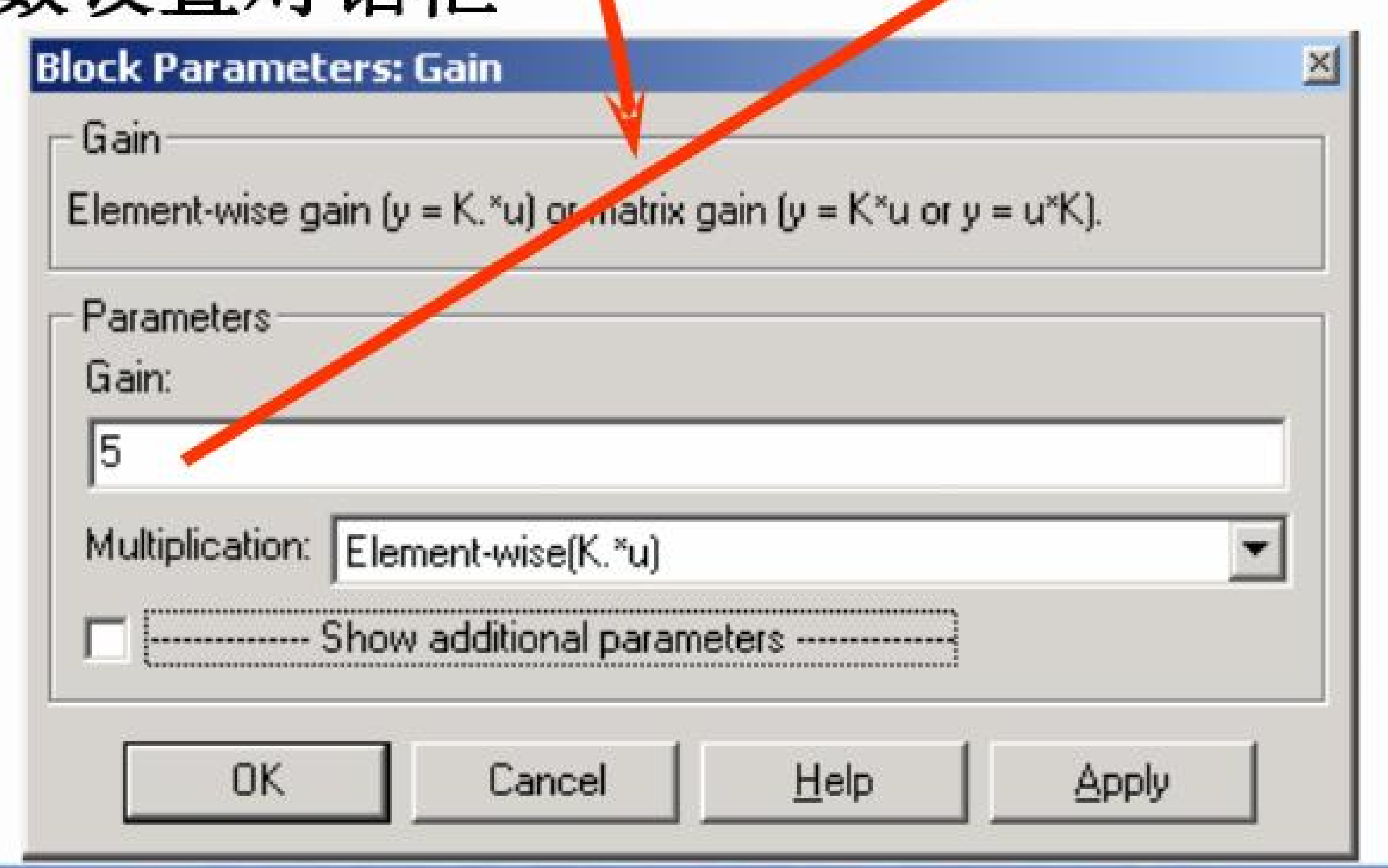
untitled *
File Edit View Simulation Format Tools Help
Normal
Sine Wave
Gain
Scope
Ready 170% ode45



设置模块参数



双击块打开参数设置对话框

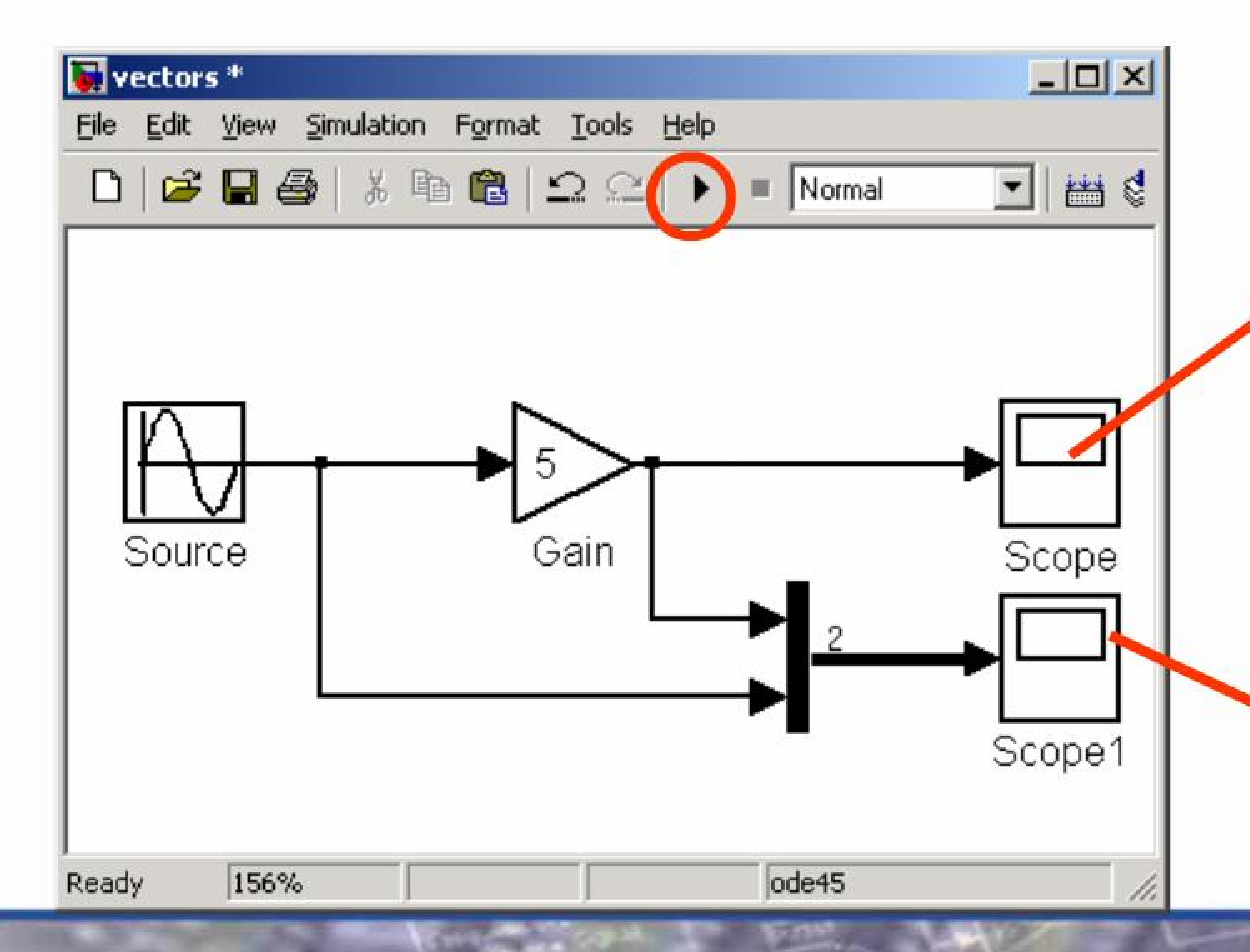


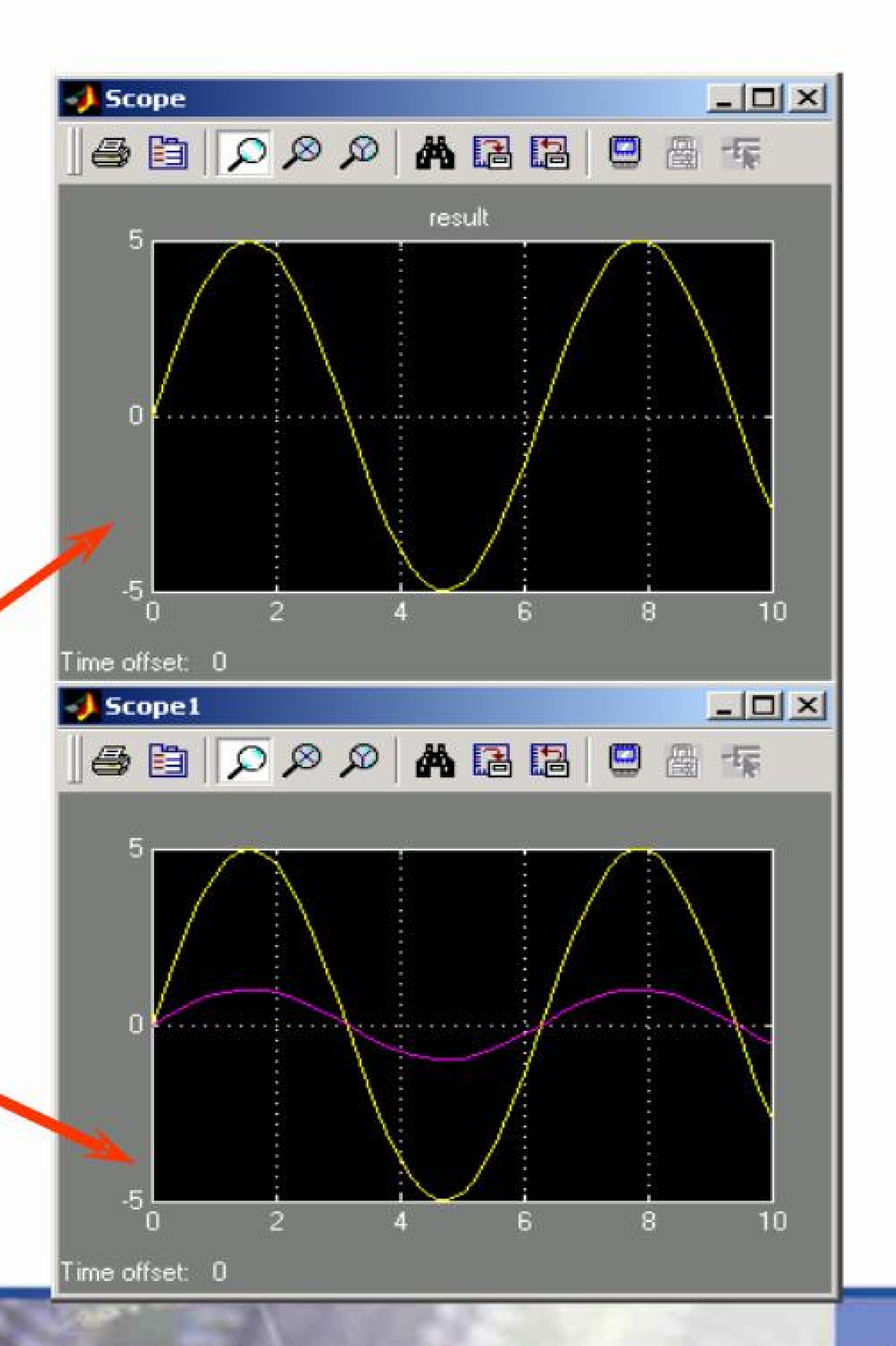
选择应用或确定按钮



仿真运行

- 点击 PLAY 按钮 运行框图仿真
- 双击示波器观察结果







无状态系统求解器选项

选择离散求解器

变步长求解器

最大允许步长

定步长值

定步长求解器

Configuration Parameters: simple_if/Configuration

Select: Solver, Data Import/Export, Optimization, Diagnostics, Sample Time, Data Integrity, Conversion, Connectivity, Compatibility, Model Referencing

Simulation time — Start time: 3 Stop time: 100

Solver options — Type: Variable-step Solver: discrete (no continuous states)

Max step size: auto

Zero crossing control: Use local settings

Configuration Parameters: simple_if/Configuration

Select: Solver, Data Import/Export, Optimization, Diagnostics, Sample Time, Data Integrity, Conversion, Connectivity, Compatibility, Model Referencing, Hardware Implementation, Model Referencing, Real-Time Workshop, Comments, Symbols, Custom Code

Simulation time — Start time: 3 Stop time: 100

Solver options — Type: Fixed-step Solver: discrete (no continuous states)

Periodic sample time constraint: Unconstrained

Fixed step size (fundamental sample time): auto

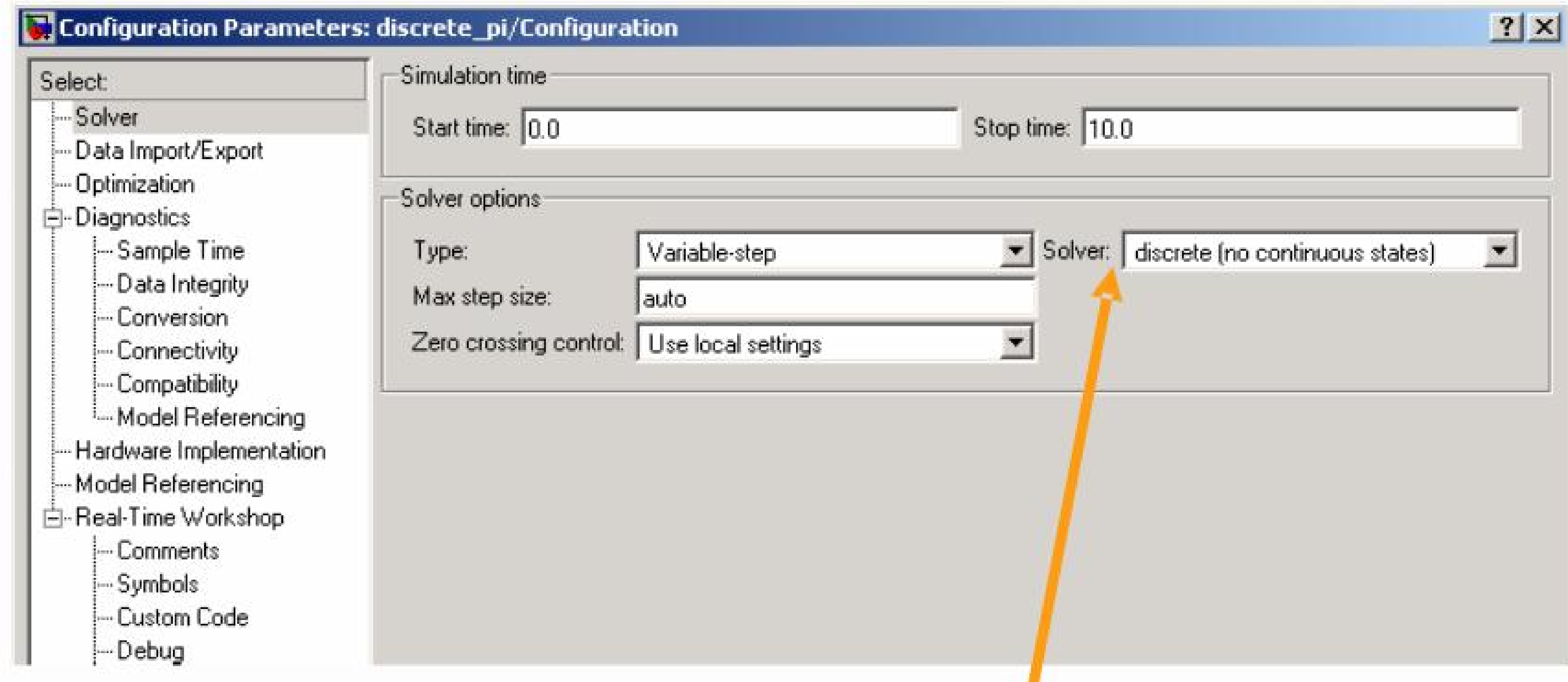
Tasking mode for periodic sample times: Auto

Higher priority value indicates higher task priority

Automatically handle data transfers between tasks

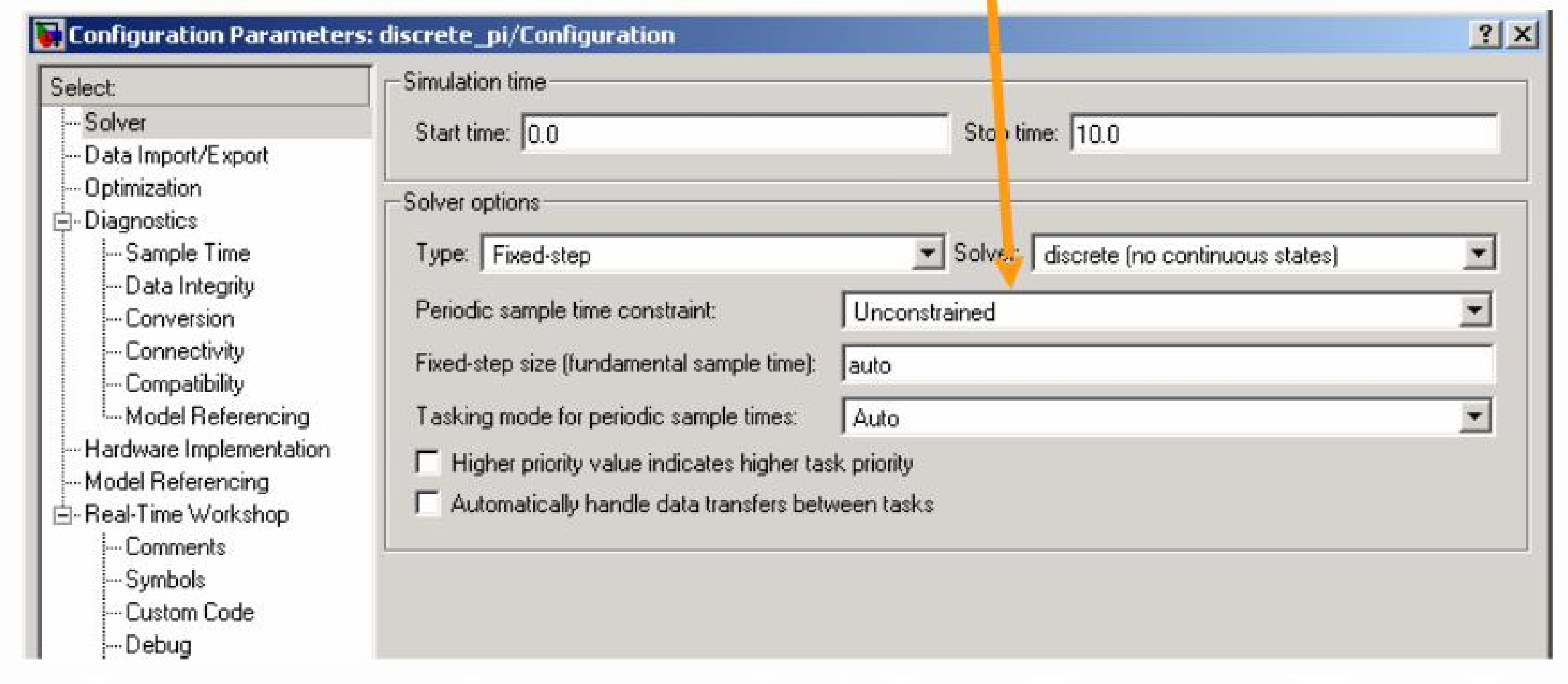


离散求解器



变步长求解器

选择离散求解器



定步长求解器



连续系统求解器选项

变步长求解器

最大允许步长

最小允许步长

初始步长

选择ODE算法

Configuration Parameters: mass_spring/Configuration

Select: Solver, Data Import/Export, Optimization, Diagnostics, Sample Time, Data Integrity, Conversion, Connectivity, Compatibility, Model Referencing, Hardware Implementation

Simulation time — Start time: 0.0 Stop time: 150

Solver options — Type: Variable-step Solver: ode45 (Dormand-Prince)
Max step size: auto Relative tolerance: 1e-3
Min step size: auto Absolute tolerance: auto
Initial step size: auto
Zero crossing control: Use local settings

定步长求解器

定步长

Configuration Parameters: mass_spring/Configuration

Select: Solver, Data Import/Export, Optimization, Diagnostics, Sample Time, Data Integrity, Conversion, Connectivity, Compatibility, Model Referencing, Hardware Implementation, Model Referencing, Real-Time Workshop, Comments

Simulation time — Start time: 0.0 Stop time: 150

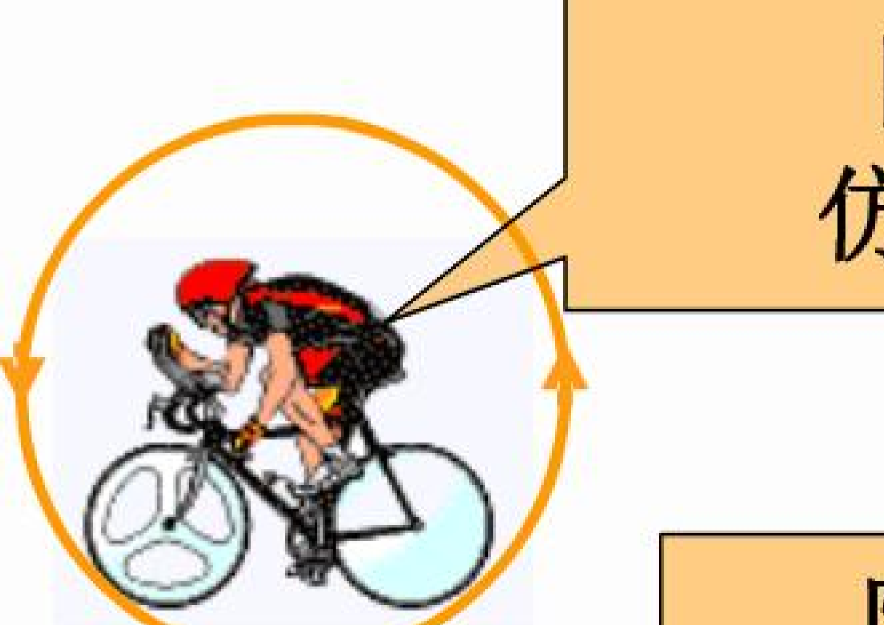
Solver options — Type: Fixed-step Solver: ode3 (Bogacki-Shanpine)
Periodic sample time constraint: Unconstrained
Fixed-step size (fundamental sample time): auto
Tasking mode for periodic sample times: Auto
Higher priority value indicates higher task priority
Automatically handle data transfers between tasks



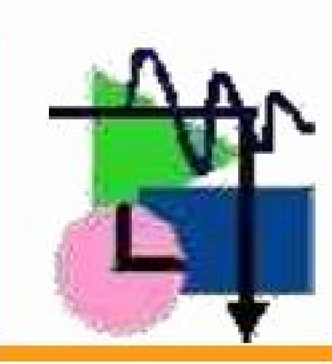
仿真过程

- 仿真阶段:
  - 初始化
  - 运行时或仿真循环
- 初始化: 1st 阶段
  - 库模块合并到模型中
  - 模型层次展开
    - 触发和使能子系统等原子子系统不展开
  - 检测信号维数宽度，数据类型和采样时间
  - 模块参数
  - 确定模型执行次序
  - 分配内存

阶段 2
仿真循环



阶段 1
初始化





仿真过程 (续)

运时: 2nd 阶段

初始化模型

计算下一个采样时间点
(仅用于变步长)

计算输出

更新离散状态

清除

计算导数

计算输出

计算导数

定位过零

仿真
(Major Step)

积分
(Minor Step)

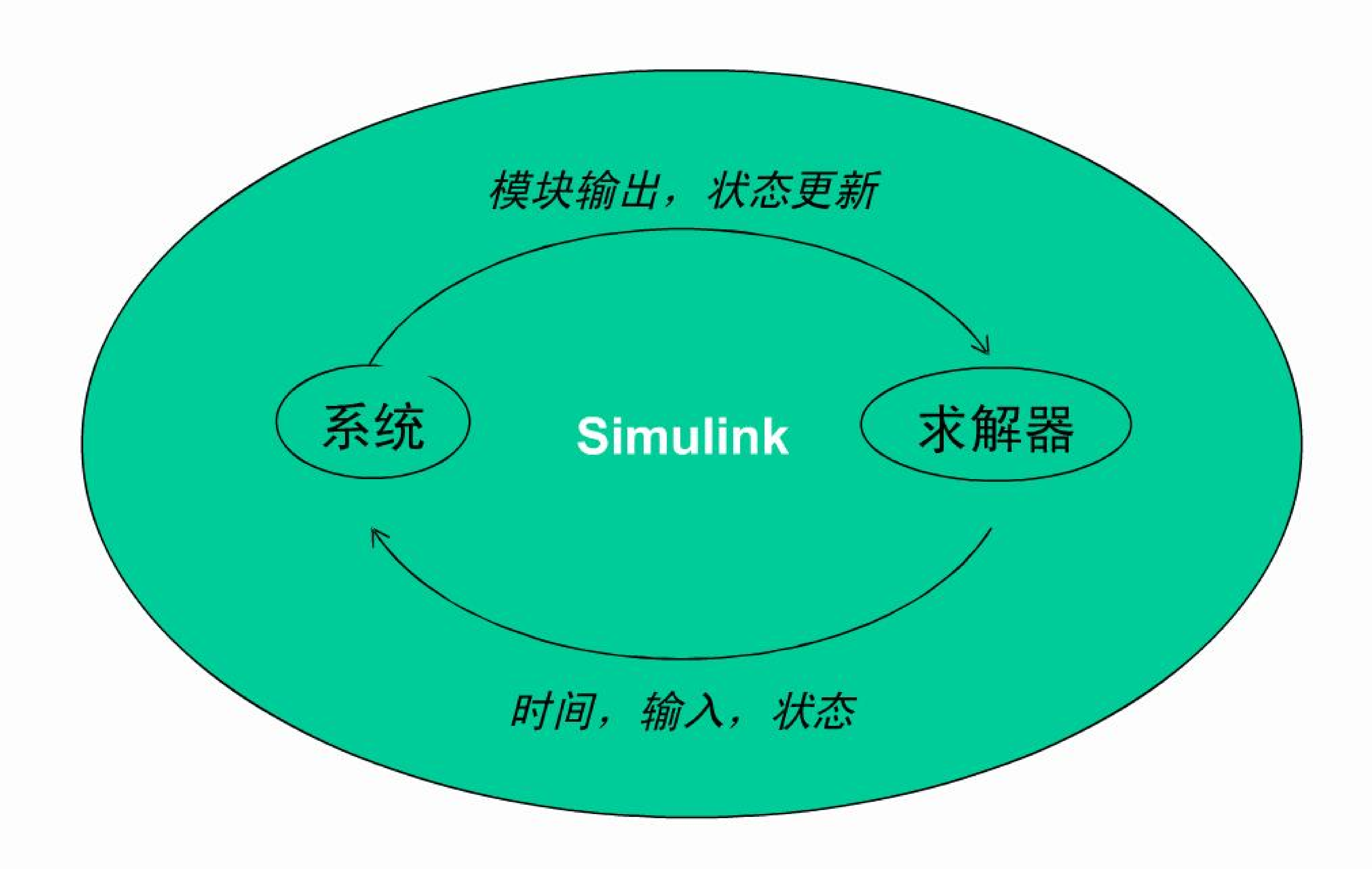
停止时间

运时/
仿真循环



求解器-系统 对话

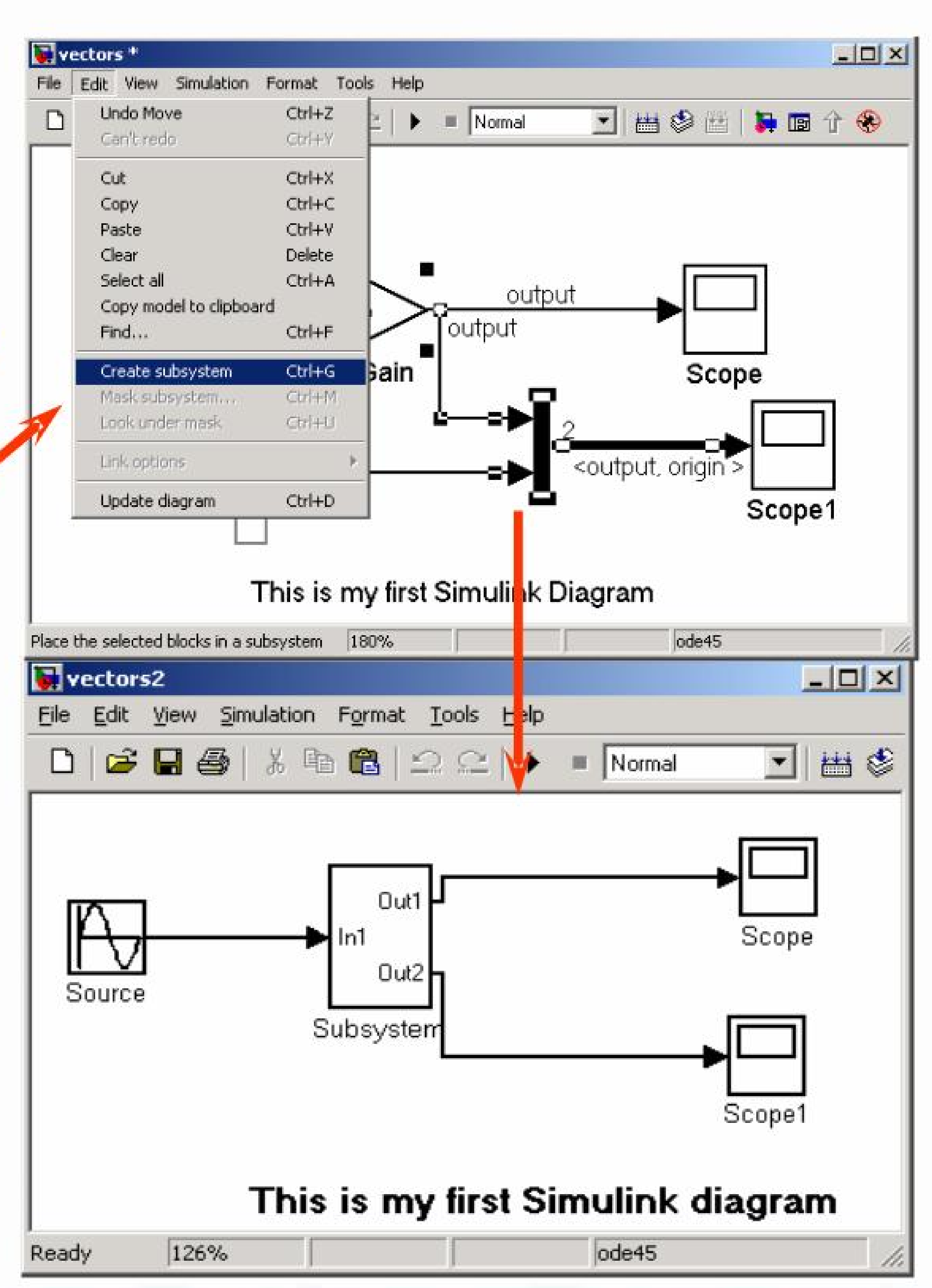
通过flags值管理

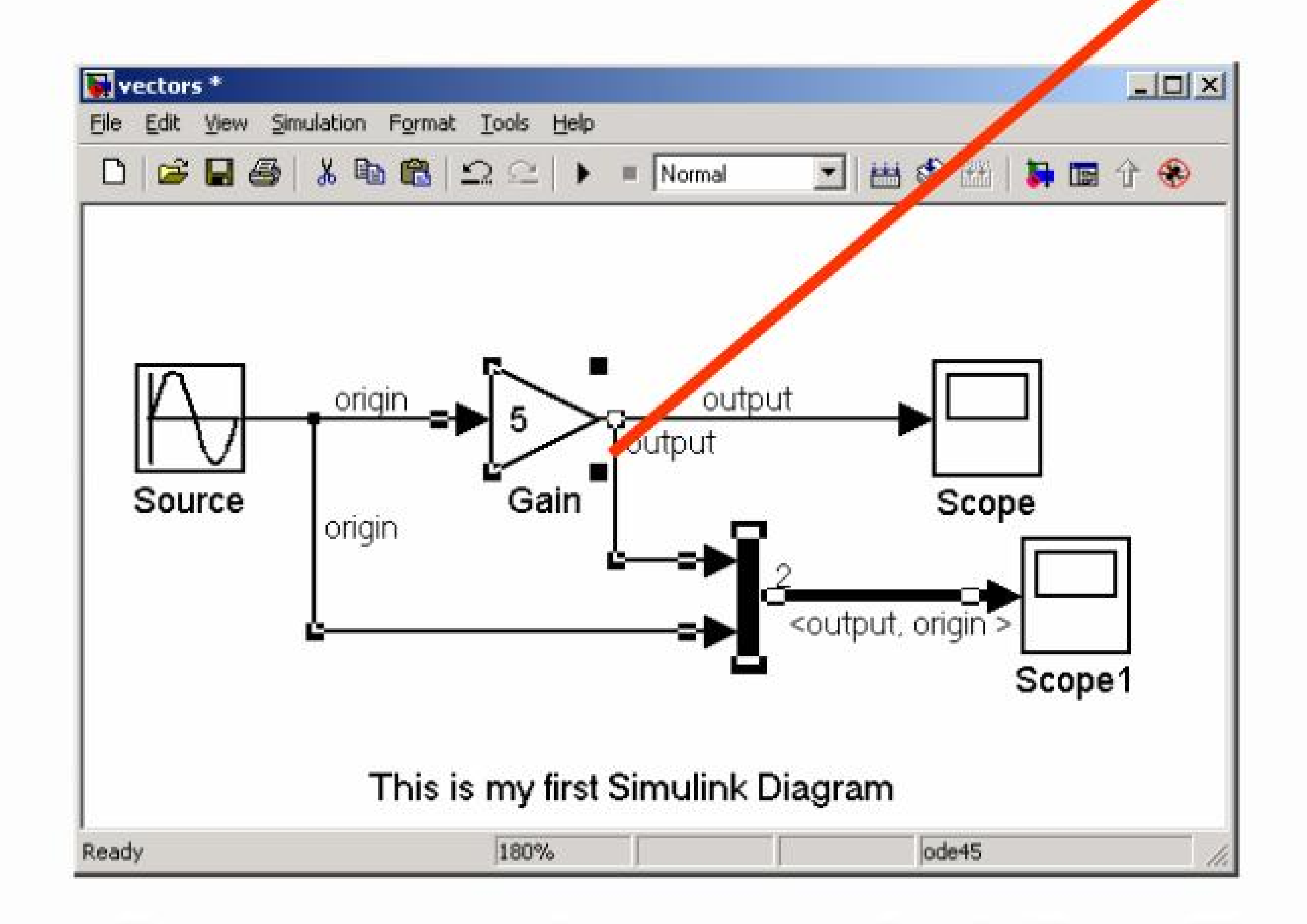




创建子系统

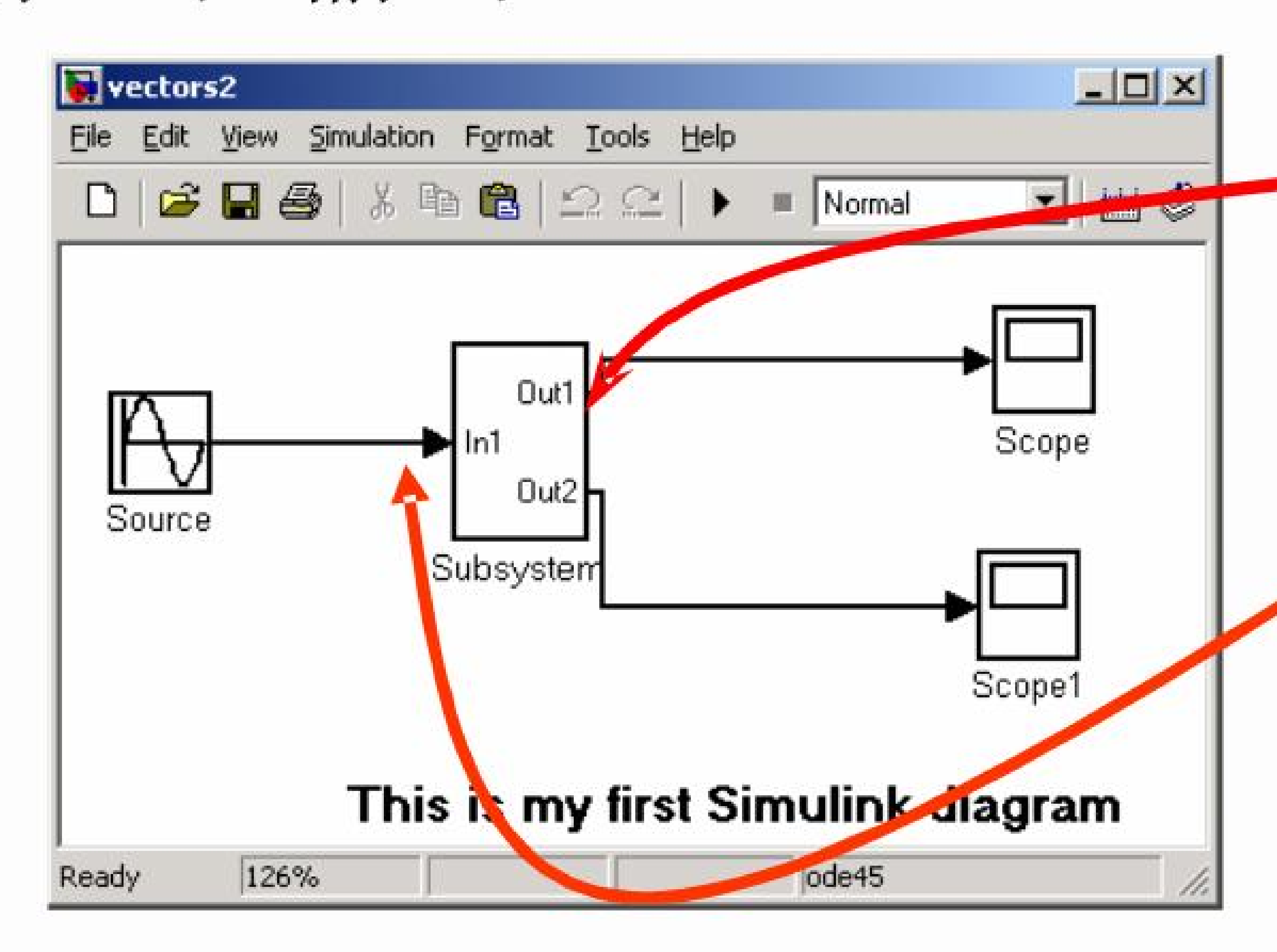
- 选择要封装的区域 (左键在背景区域上拖放)
- 在Edit菜单下选择 :“Create Subsystem”

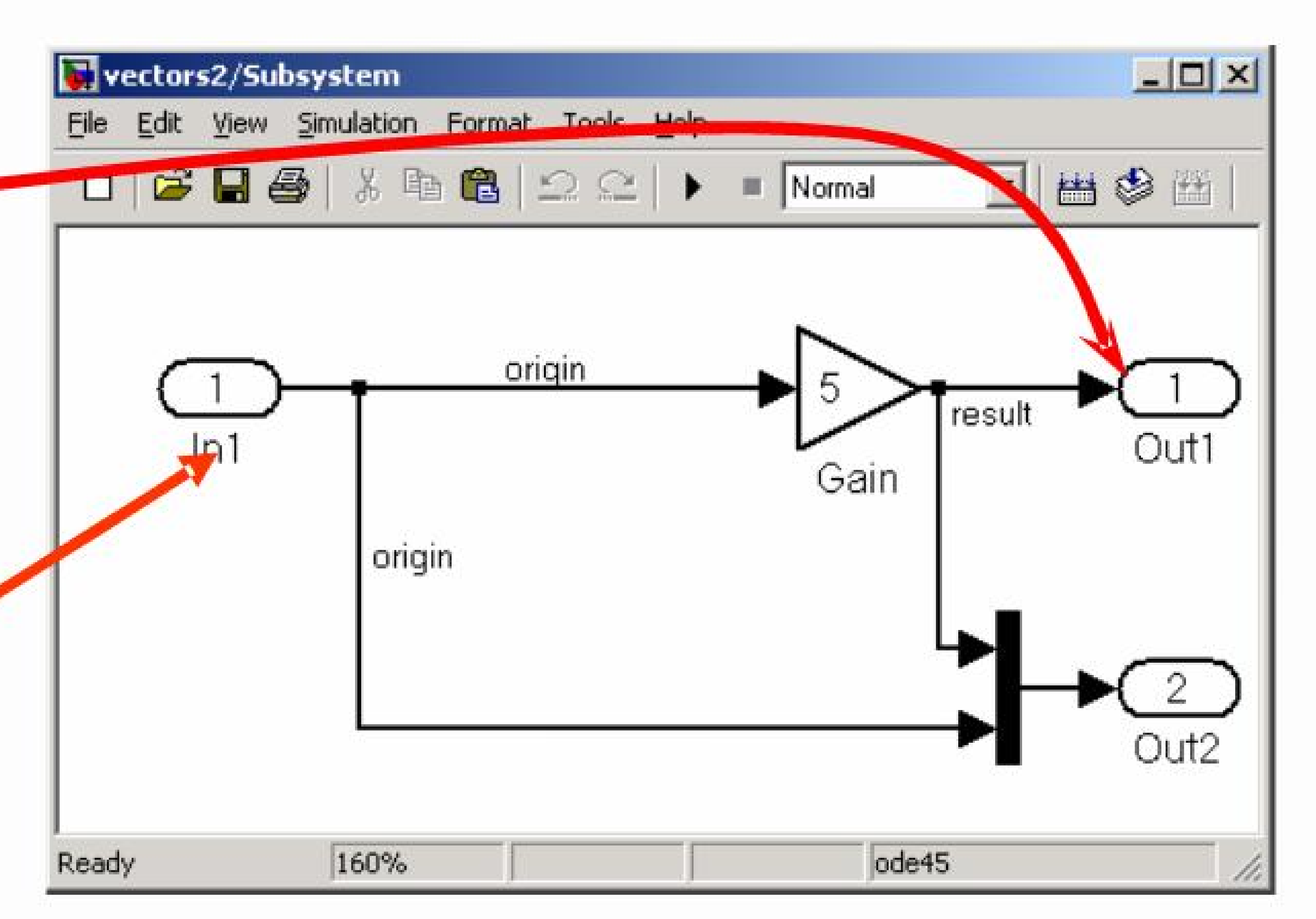






输入和输出





- 使信号可以在子系统和父系统之间传递
- 输入和输出模块的名字反映在子系统上
- 顺序标号



封装子系统

- 自定义块并带有自定义图标
- 用户双击图标时显示对话框
- 可以给自定义块加帮助
- 有自己的工作区

这样能够帮助用户

- 向子系统中传递参数
- 屏蔽不需要用户看到的细节
- “隐藏”那些不需要过多展现的内容
- 保护块的内容被那些头脑简单的人的篡改



封装一个子系统

population_grouped
File Edit View Simulation Format Tools Help
Normal
Out1
population dynamics
Ready 139%

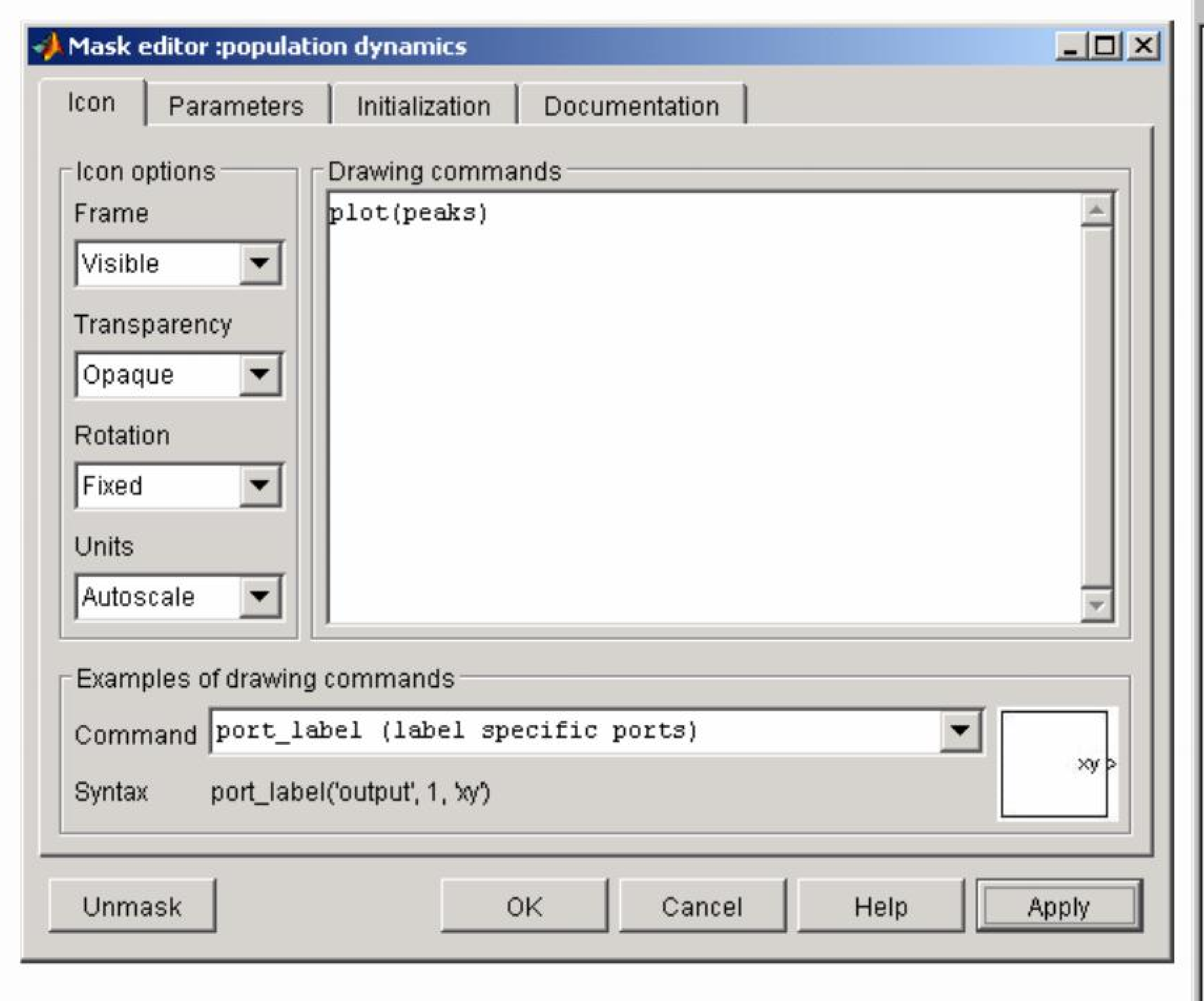
Mask editor :population dynamics
Icon | Parameters | Initialization | Documentation
Icon options
Frame: Visible
Transparency: Opaque
Rotation: Fixed
Units: Autoscale
Drawing commands
Examples of drawing commands
Command: port_label (label specific ports)
Syntax: port_label('output', 1, 'xy')
Unmask | OK | Cancel | Help | Apply

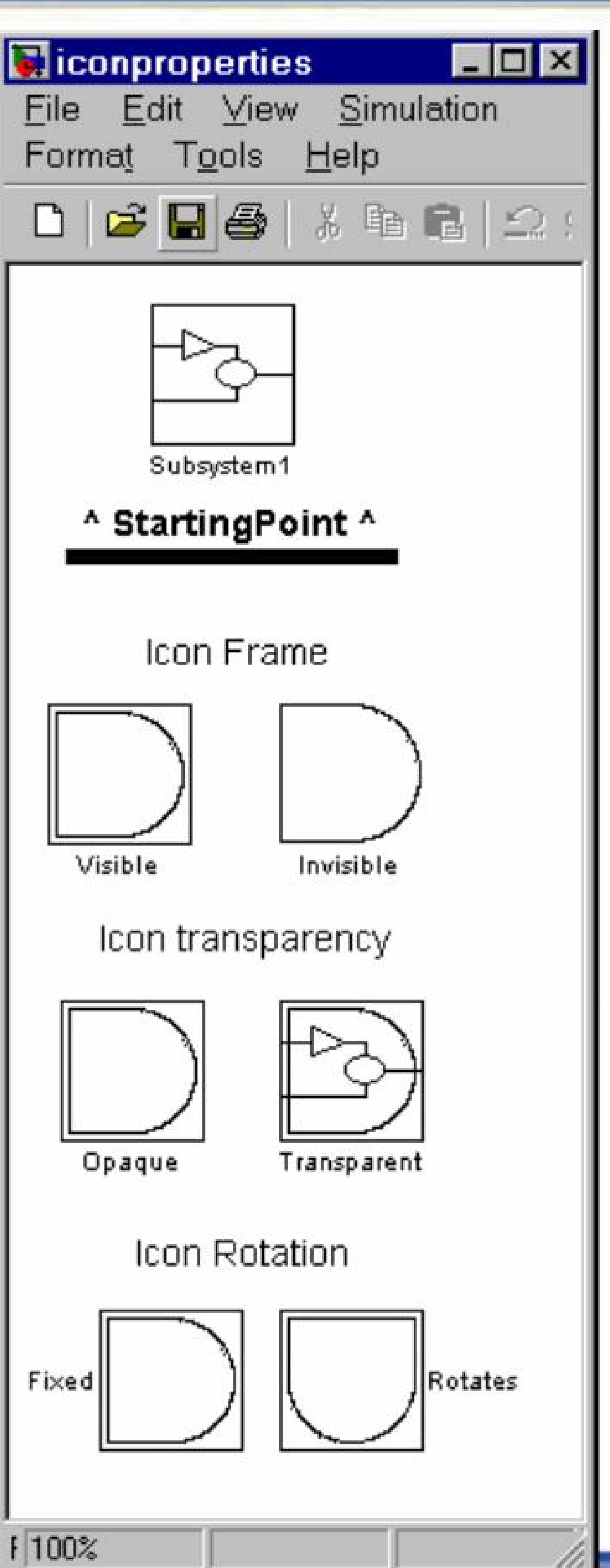
- 右键单击子系统，选择 'Mask Subsystem'

```
>> population_grouped
```


恒润科技
HIRAIN TECHNOLOGIES

图标编辑器

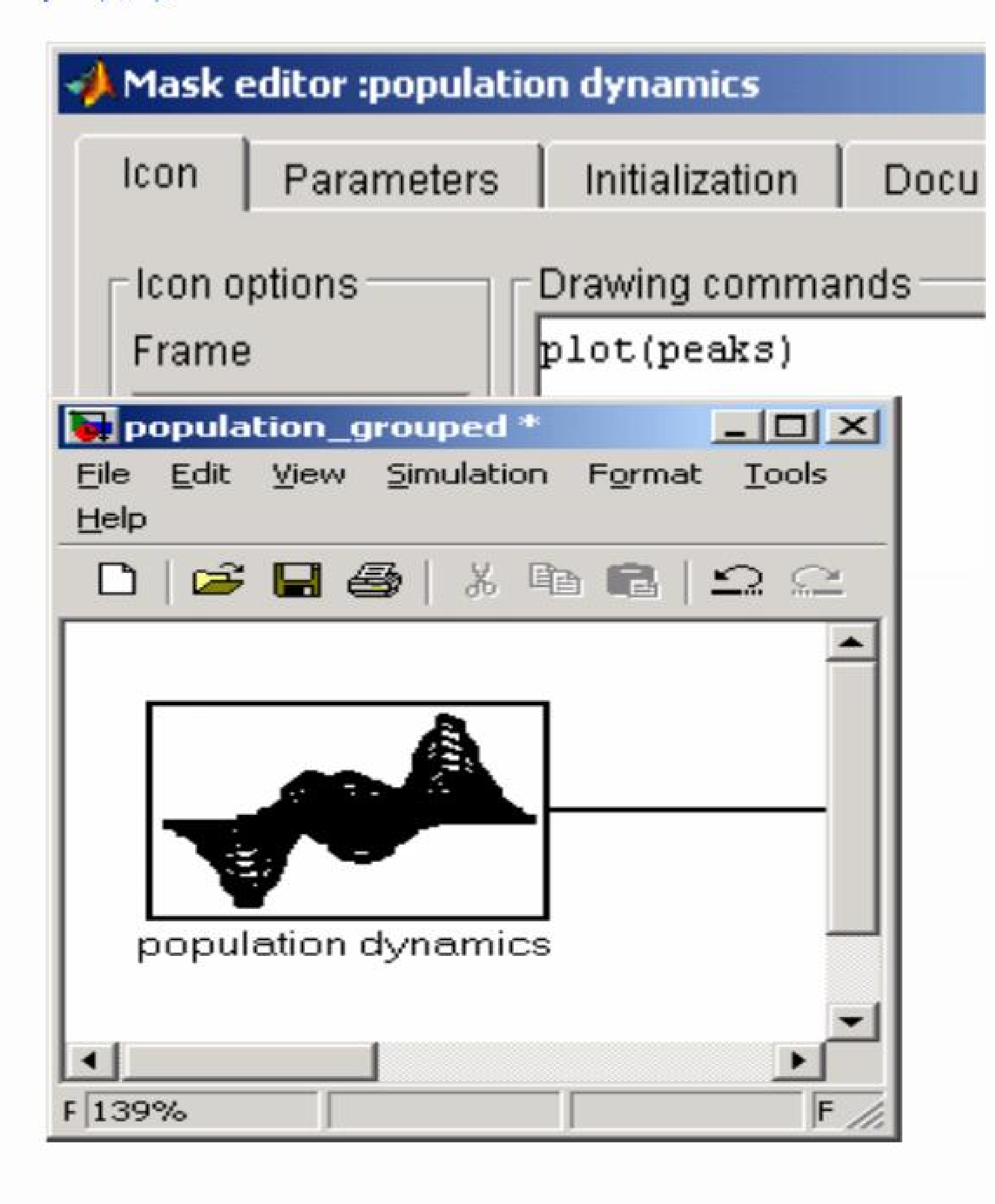

Mask editor :population dynamics
Icon
Parameters
Initialization
Documentation
Icon options
Frame
Visible
Transparency
Opaque
Rotation
Fixed
Units
Autoscale
Drawing commands
plot(peaks)
Examples of drawing commands
Command
port_label (label specific ports)
Syntax
port_label('output', 1, 'xy')
xy
Unmask
OK
Cancel
Help
Apply


iconproperties
File
Edit
View
Simulation
Format
Tools
Help
Subsystem1
^ StartingPoint ^
Icon Frame
Visible
Invisible
Icon transparency
Opaque
Transparent
Icon Rotation
Fixed
Rotates
100%

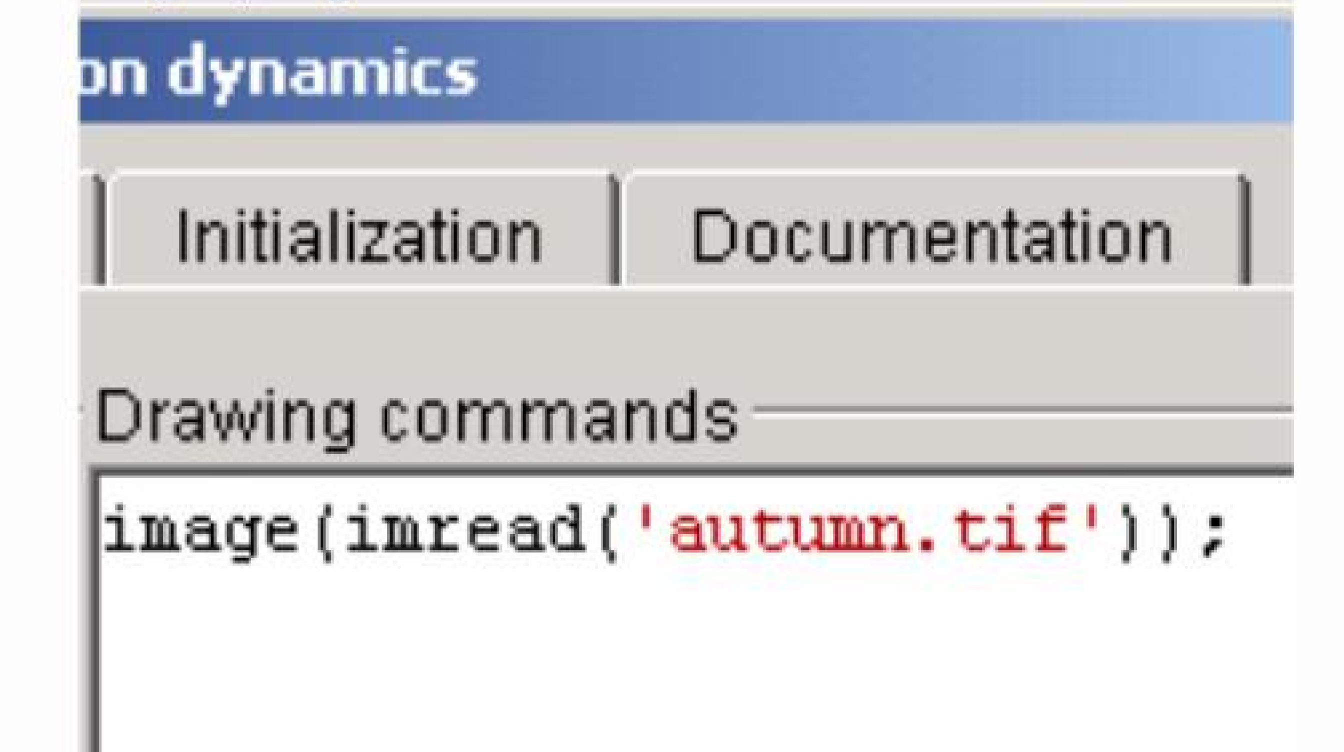


一些图标

- 图形



- 图象



- 文本

on dynamics

Initialization | Documentation

Drawing commands

```
disp('This is my subsystem');
```



对话框参数

Mask editor :population dynamics

Icon | Parameters | Initialization | Documentation

Dialog parameters

Prompt	Variable	Type	Evaluate	Tunable
Resource Limit	K	edit	☑	☑
Reproduction Rate	r	edit	☑	☑

Options for selected parameter

Popups (one p

Unmask | Ok

Block Parameters: population dynamics

Subsystem (mask)

Parameters

Resource Limit

1e6

Reproduction Rate

1.05

OK | Cancel | Help | Apply

- 封装子系统的工作空间
  - MATLAB工作空间的参量不再可见
- 传递参量给子系统



封装过的工作区

- 子系统可以比作脚本
  - 没有输入参数
  - 使用 MATLAB 工作区中的变量
- 封装过的子系统可以比作函数
  - 通过对话框输入参数
  - 不能使用 MATLAB 工作区变量
  - 它包含的变量对于其他子系统不可见
- 通过分离工作区，用户可以在同一个模型中使用多个同样的子系统，对于同一个变量它们可以有不同的值。



初始化命令

Mask editor :population dynamics

Icon | Parameters | Initialization | Documentation

Dialog variables: K, r

Initialization commands

```
str1 = ['K = ',num2str(K)];
str2 = ['r = ',num2str(r)];
icon = strvcat(str1,str2);
```

Allow library block to modify its contents

Unmask | OK

Mask editor :population dynamics

Icon | Parameters | Initialization | Documentation

Icon options: Frame: Visible; Transparency

Drawing commands

```
disp(icon);
```

- 初始化命令是一般的MATLAB 命令
  - 当块对话框或者仿真开始时这个命令的改变有效



文档

Mask editor :population dynamics

Icon | Parameters | Initialization | Documentation

Mask type

population

Mask description

Population Dynamics
p(k) = r*p(k-1)*(1-(p(k-1)/K)

Mask help

This block implements simple <i> population </i> dynamics

Unmask | OK | Cancel | Help | Apply

Block Parameters: population dynamics

population (mask)

Population Dynamics
p(k)=rp(k-1)(1-p(k-1)/K)

Parameters

Resource Limit

Reproduction Rate

OK | Cancel | Help | Apply

Find in page: Go

Simulink Masked Block: population

Using Simulink

population

This block implements simple *population* dynamics

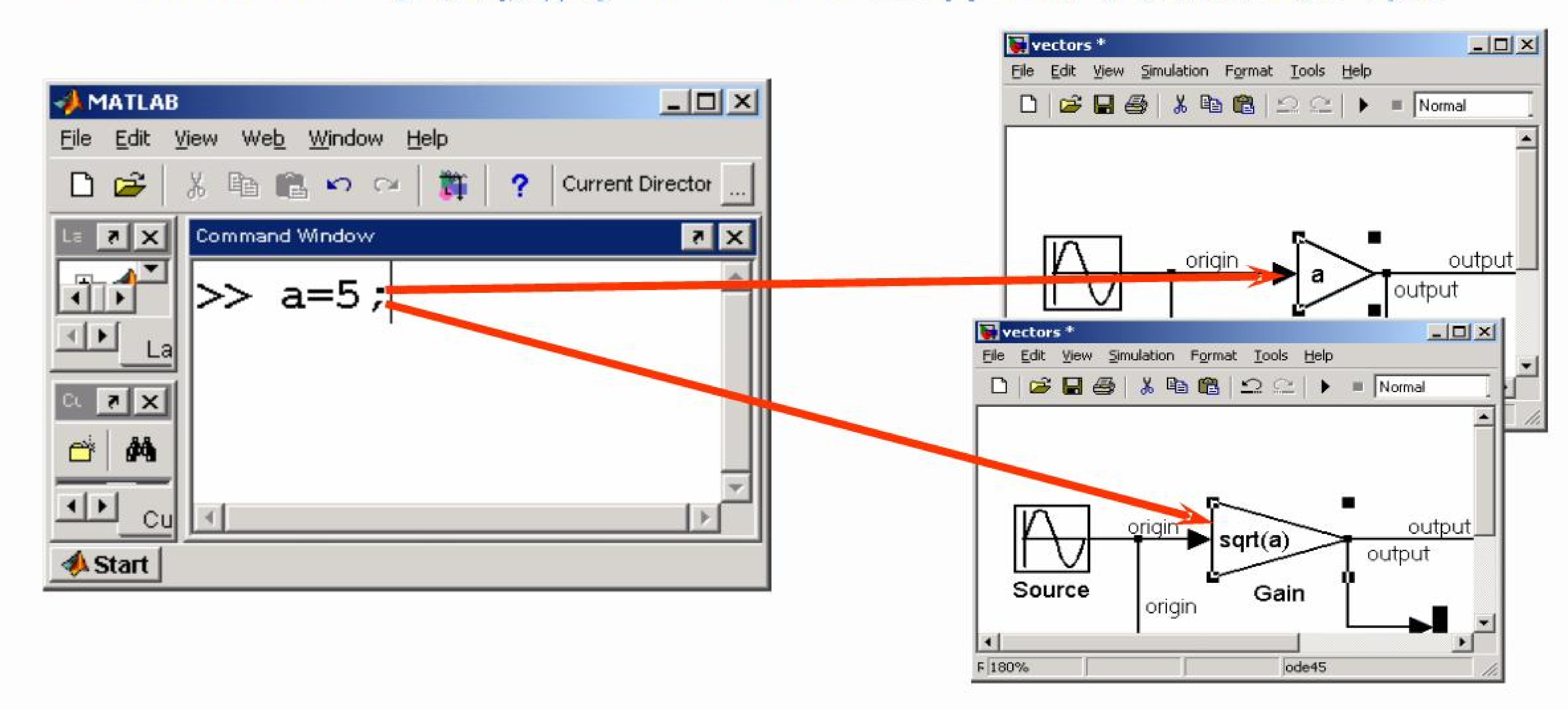
HTML Text

>>population_masked



来自工作区的参数

- Simulink 能够使用 MATLAB 的工作区
- 模块参数可以是来自 MATLAB 工作区
- Simulink可以使用MATLAB工作区对表达式赋值





将信号输出到工作区

- Simulink 能够使用 MATLAB 工作区输入/输出数据.
- 使用“To Workspace” 模块.
- 当仿真暂停或终止时输出将被写到工作区中.
- 可以保存为结构（包括时间）或数组

```
>> outwork
```

outwork
File Edit View Simulation Format Tools Help
Normal
Sine Wave
a
Gain
simout
To Workspace

MATLAB
File Edit View Web Window Help
Command Window

```
>> a=5;
>> whos
  Name

  a
  simout
  tout

Grand total is

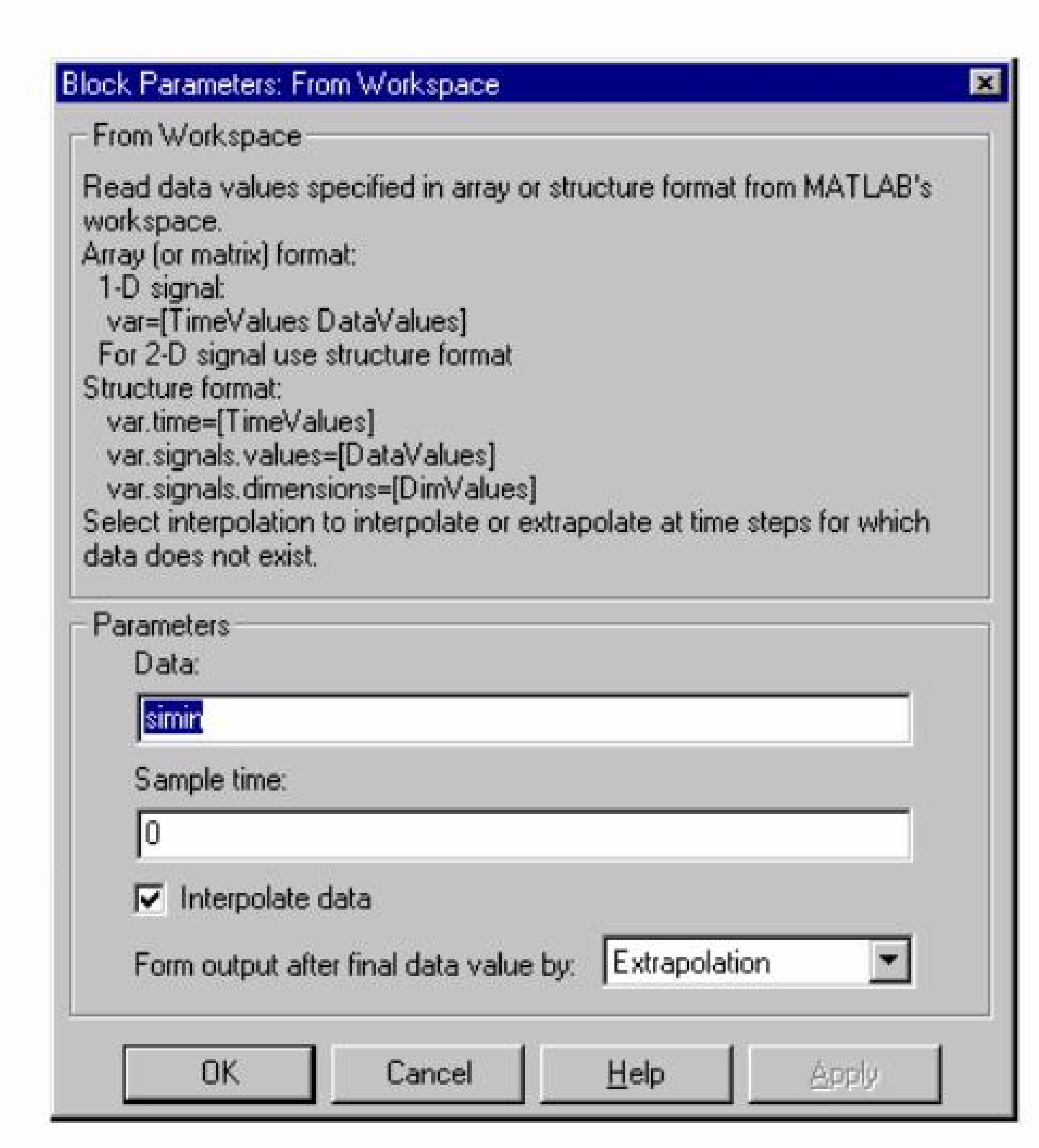
>>
```

Block Parameters: To Workspace
To Workspace
Write input to specified array or structure in MATLAB's main workspace.
Data is not available until the simulation is stopped or paused.
Parameters
Variable name: simout
Limit data points to last: inf
Decimation: 1
Sample time (-1 for inherited): -1
Save format: Structure
OK Cancel Help Apply



从工作区输入信号

- From Workspace 块可以将 MATLAB 工作区中的变量作为输入变量
- 格式:
  - t=0:time_step:final_time;
  - x=func(t);或分段线性插值
  - 块的参数: [t’，x’]



- 例子:

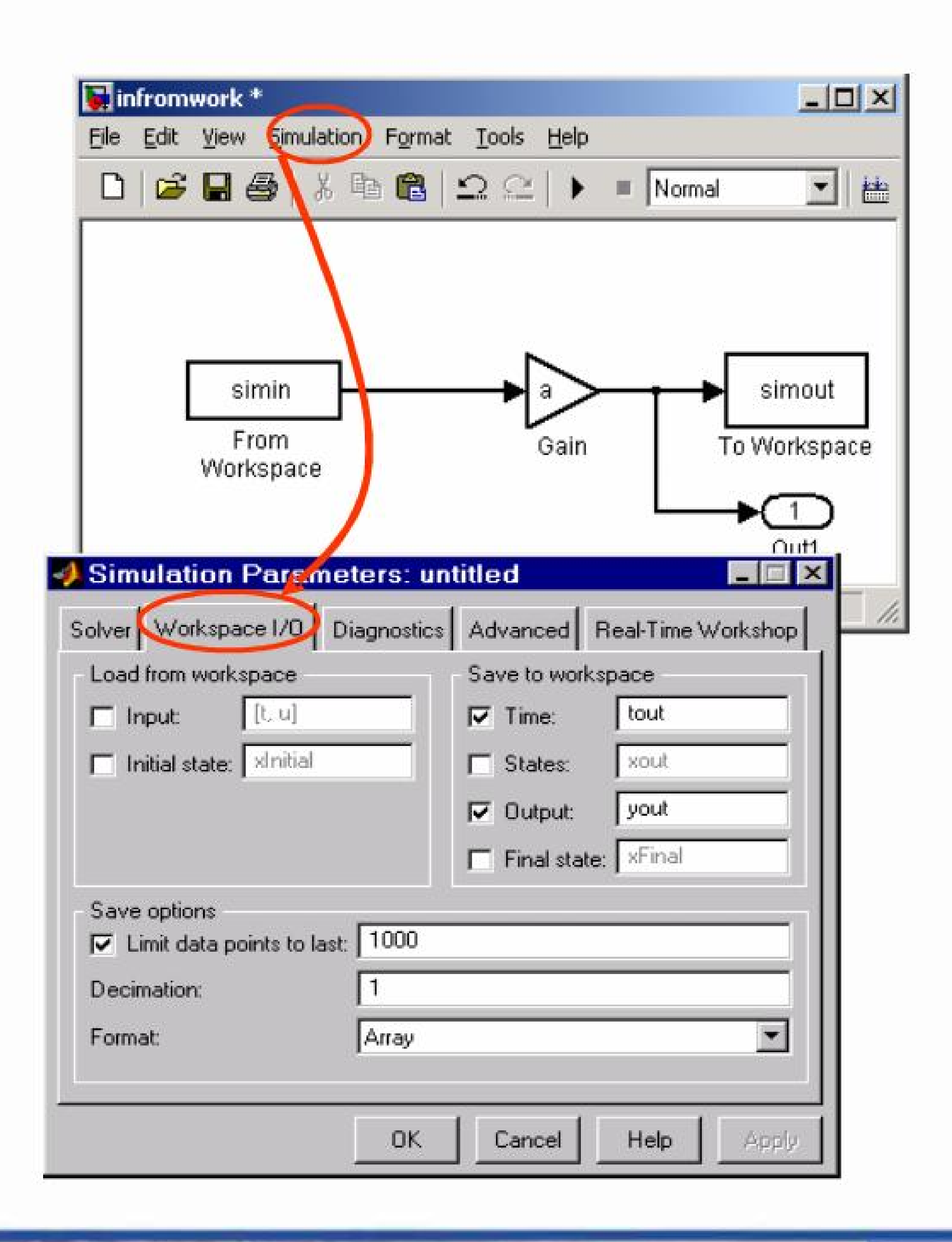
```
>> infromwork
>> t=0:0.1:10; x=sin(t);
>> t=[0 3 6 9]; x=[-1 1 -1 1];
>> simin=[t’, x’];
```



工作区 I/O 和 Inport/Outport 输入，输出

- 将Inport/ Outport 置于系统顶层 ==> 将数据传入/传出工作区
- 数据格式与 To/From Workspace 块相同
- 选项控制

  Simulation--> Simulation Parameters--> Workspace I/O Tab





为什么通过命令行仿真?

- 自动地重复运行仿真
- 参数调整
- 分析不同输入下的响应
- 快速仿真



基本命令行语法

[t,x,y]=sim(model)

[t,x,y_1,y_2,…, y_n]=sim(model)

- 使用模型中的所有参数仿真模型
- 时间，状态和输出在仿真后返回
- 可得到单独的输出
- 输出从模型的最顶层的 outport 块得到

练习：

```
>> in_out
>> [t,x,y]=sim('in_out');
```



使用命令行 – 时间

```
[t,x,y]=sim(model,tFinal)
[t,x,y]=sim(model,[tStart tFinal])
[t,x,y]=sim(model,[tStart outputTimes tFinal])
```

- 除了结束时间外，使用模型中的所有参数对模型进行仿真
- 也可以替代开始和输出时间

```
>> [t,x,y]=sim('in_out',5);
>> [t,x,y] = sim('in_out',[1 5]);
>> [t,x,y] = sim('in_out',[1 3 5 7 9]);
```



使用命令行 - 外部输入

```
[t,x,y]=sim(model, timespan, options, tu)
[t,x,y]=sim(model, timespan,[], tu)
[t,x,y]=sim(model,[],options, tu)
[t,x,y]=sim(model,[],[], tu)
```

- tu 允许为模型的最顶层的 inport 提供外部输入
- 处于时间或选项位置的 [ ] 表示采用当前模型中的设置

```
>> [t,x,y]=sim('in_out',50,[],ut);
```



仿真选项

[t,x,y]=sim(model, timespan, options)

- 选项为一个数据结构，包含求解器所有参数
- 选项结构覆盖所有仿真参数设置

```
» options=simget('pendulum')

options =

          AbsTol: 1.0000e-006
           Debug: 'off'
      Decimation: 1
    DstWorkspace: 'current'
  FinalStateName: ''
       FixedStep: 'auto'
    InitialState: []
     InitialStep: 'auto'
        MaxOrder: 5
      SaveFormat: 'Matrix'
         MaxRows: 0
         MaxStep: 0.1000
    OutputPoints: 'all'
 OutputVariables: ''
          Refine: 1
          RelTol: 0.0010
          Solver: 'ode45'
    SrcWorkspace: 'base'
           Trace: ''
       ZeroCross: 'on'
```

> options=simget('modelname');

> newoptions=simset('OptionName', new_value);

使用 **simget** 和 **simset** 来提取和设置选项结构。



使用框图属性

- 框图应处于打开状态（使用load_system调入内存）
- 使用GET_PARAM 确定框图属性的值

```
>> get_param('bungee','AbsTol')
```

- 使用SET_PARAM 改变框图属性的值

```
>> set_param('bungee','AbsTol','1e-3')
```

- 也可以获得/修改块的属性

```
>> set_param('bungee/Integrator IC=-30','AbsTol','1e-3')
```

- 这些属性与模型一起被保存



Model Explorer

```
>> model9
```



小结

- Simulink基础
- 模块的操作与参数的设置
- 仿真选项的设置和仿真机理
- 创建子系统与封装子系统
- Simulink 与MATLAB 工作区的接口
- 通过命令行仿真
- 模型浏览器Model Explore



课程概要

- Simulink建模仿真
- 面向DSP的自动代码生成
- 面向FPGA的自动代码生成



本章概述

- 实时程序的构架
- 代码格式
- GRT target 的目的
- 生成简单系统 GRT 目标的代码
  - GRT 代码生成选项
- Build 的过程 – 对比
- 生成的文件和代码结构概况
- 模块的结构
- 实时数据的结构
- 示例：生成离散和连续系统模型的代码
- 配置并生成 ERT 目标的代码
  - ERT 目标的代码生成选项
- 代码生成的过程(阶段)
- 信号记录



本节 将比较 GRT 和 ERT 目标

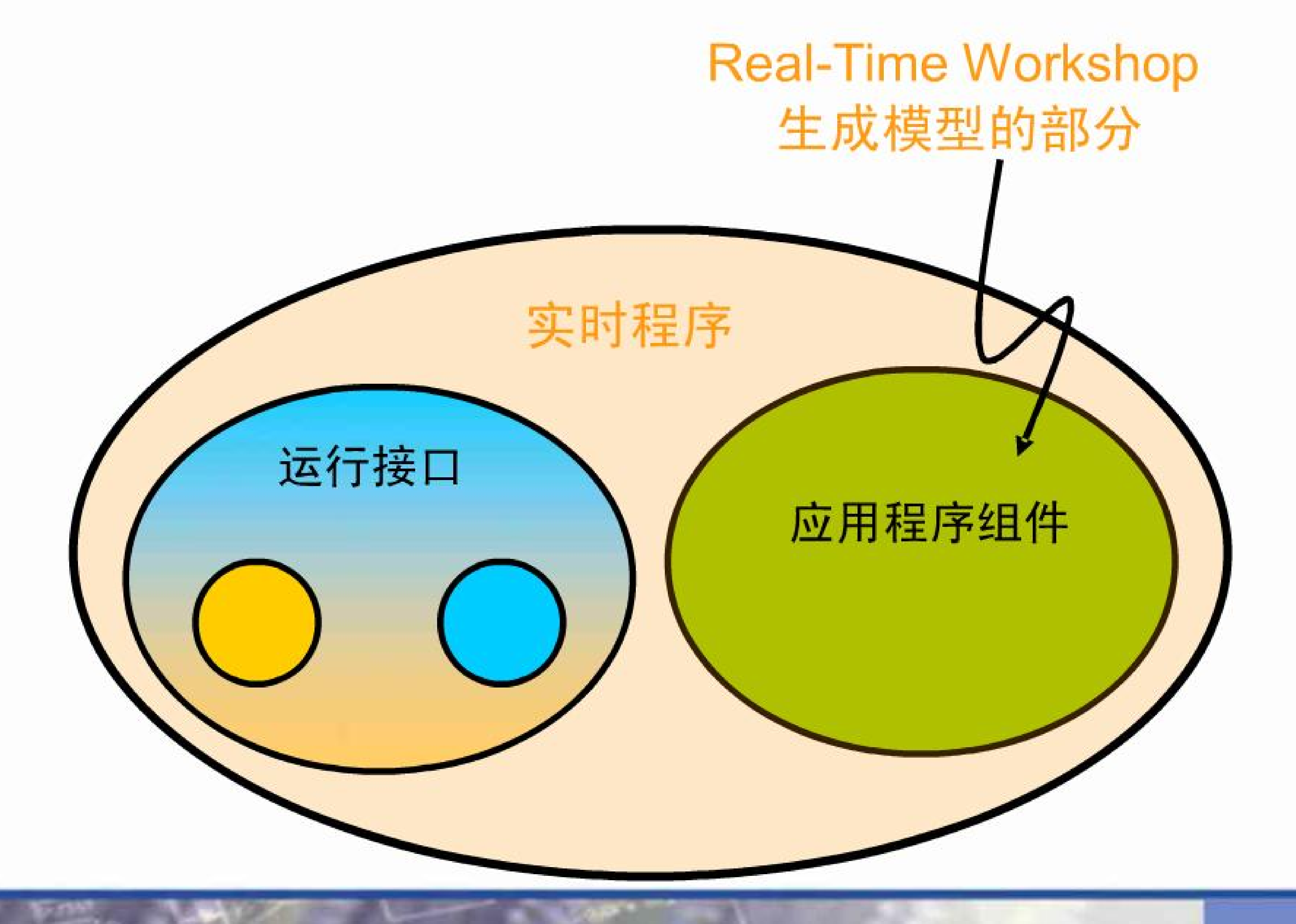


实时程序的构架

- 一个实时程序由两个主要部分组成：
  - 运行接口(RTI)
  - 应用程序组件

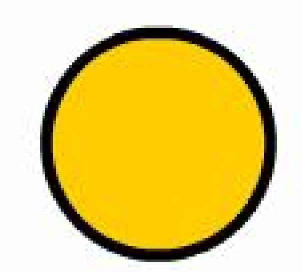
与系统/环境有关的组件

与系统/环境无关的组件





实时程序的构架(续)



与系统/环境相关的组件(targetname_main.c)

(控制程序的定时，创建任务，安装中断管理，开启数据记录和进行错误检查)

初始化

- 初始化数值参数
- 初始化实时模型部分和S-functions
- 采样时间和偏移量初始化
- 调用 MdlStart并初始化状态
- 设定控制模型执行的定时器
- 初始化定时引擎
- 定义后台任务和开启书据记录(如果选择的话)

模型运行

- 运行后台任务(外部模式)或等待下一个采样时间
- 检查采样时间和任务ID
- 执行模型 – 调用并运行时间函数(由中断启动)
- 将数据记录到缓冲区
- 从中断返回

程序结束

- 结束运行
  - 清除实时模型结构
  - 释放内存
  - 将数据写入文件

- targetname 可以是 grt, ert, rsim 等等。
- targetname_main.c 包含主函数和运行函数



实时程序的构架(续)

与系统/环境无关的组件
(rt_sim.c – 实现程序的定时引擎
ode#.c - 实现数值积分算法)

应用程序/
模型专属组件

模型运行

从硬件读取数据

计算系统输出

向硬件写输出

决定下一个采样时间点和任务 TaskID

更新离散状态

计算导数

时间增量

定时引擎
(rt_sim.c)

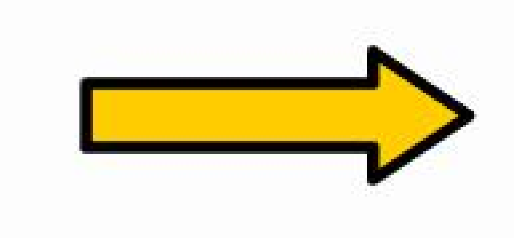
ode#.c

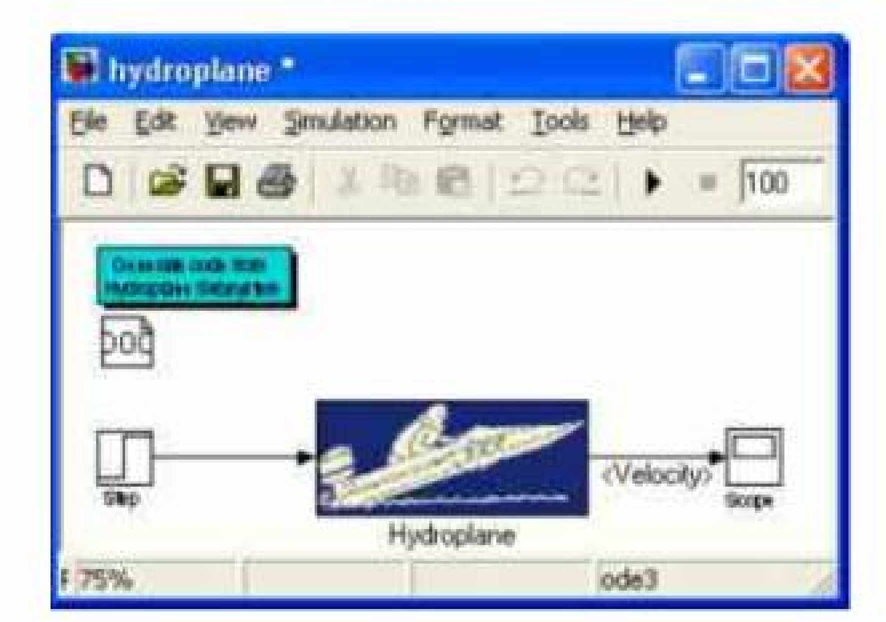
定时引擎
(rt_sim.c)



实时程序的构架(续)

应用程序/模型
特定组件





- 包括来自方框图和 C MEX S-Functions 的代码
- 在.mdl 模型文件所在的位置实现 model.c
- 功能：
  - 初始化实时模型，采样时间和偏移量
  - 计算模块和系统输出
  - 更新离散状态向量
  - 计算连续模型状态量的导数
  - 程序结束时有条理的结束
  - 收集模块和示波器的书据并作记录



实时程序的构架(续)

实时运行

与系统/环境相关的组件

主程序：`targetname_main.c`
定时
中断管理
I/O 驱动
数据记录

后台任务和数据交换
`ext_svr.c`, ...

与系统/环境无关的组件

求解器：`ode1.c` ... `ode5.c`
定时引擎：`rt_sim.c`
实时对象

应用程序组件

生成的代码：`model.c`， ...
`mdlOutputs()`, ...
内嵌 S-functions
模型参数：`model_data.c`

非内嵌
S-functions
`myfunction.c`



代码格式

- 每个目标使用独特的代码格式
- 选择目标的同时也隐含选择了代码的格式
- Real-Time Workshop 中有以下几种可用的格式
  - S-function/Accelerator 代码格式
    - C MEX S-function 结构
  - Real-time代码格式
    - 监测和调参能力
    - 内存静态分配
  - Real-time malloc代码格式
    - 动态分配内存
    - 支持同一模型的多个实例
  - Embedded 代码格式
    - 简单的调用接口，占用存储空间少



使用 GRT 目标的目的

GRT 目标

- 转向用户定制快速原型目标的起点
- 使用一种实时代码格式
- 支持外部模式(external mode)通讯
- 如果要求更高的内存使用效率，可以使用Real-Time malloc 代码格式。

与 ERT 目标相比较

- 目标环境 Real-Time Workshop Embedded Coder
- 转向用户定制嵌入式应用程序的起点
- 使用嵌入式 C 代码格式
- 可编辑接口文件，配合用户的特定需要



配置目标

Configuration Parameters: simplegain/Configuration

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Integrity
  - Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing
- Real-Time Workshop
  - Comments
  - Symbols
  - Custom Code
  - Debug
  - Interface

Target selection

RTW system target file: grt.tlc   Browse...

Description: Generic Real-Time Target

Documentation

☑ Generate HTML report

☑ Launch report after code generation completes

Build process

TLC options:

Make command: make_rtw

Template makefile: grt_default_tmf

☑ Generate code only

System target file browser: simplegain

system target file	description
asap2.tlc	ASAM-ASAP2 Data Definition Target
ert.tlc	RTW Embedded Coder (no auto configuratic
ert.tlc	RTW Embedded Coder (auto configures for
ert.tlc	RTW Embedded Coder (auto configures for
ert.tlc	Visual C/C++ Project Makefile only for t
grt.tlc	Generic Real-Time Target
grt.tlc	Visual C/C++ Project Makefile only for t
grt_malloc.tlc	Generic Real-Time Target with dynamic me
grt_malloc.tlc	Visual C/C++ Project Makefile only for t

Full name: D:\Applications\matlab7lcs\rtw\c\grt\grt.tlc

Template make file: grt_default_tmf

Make command: make_rtw

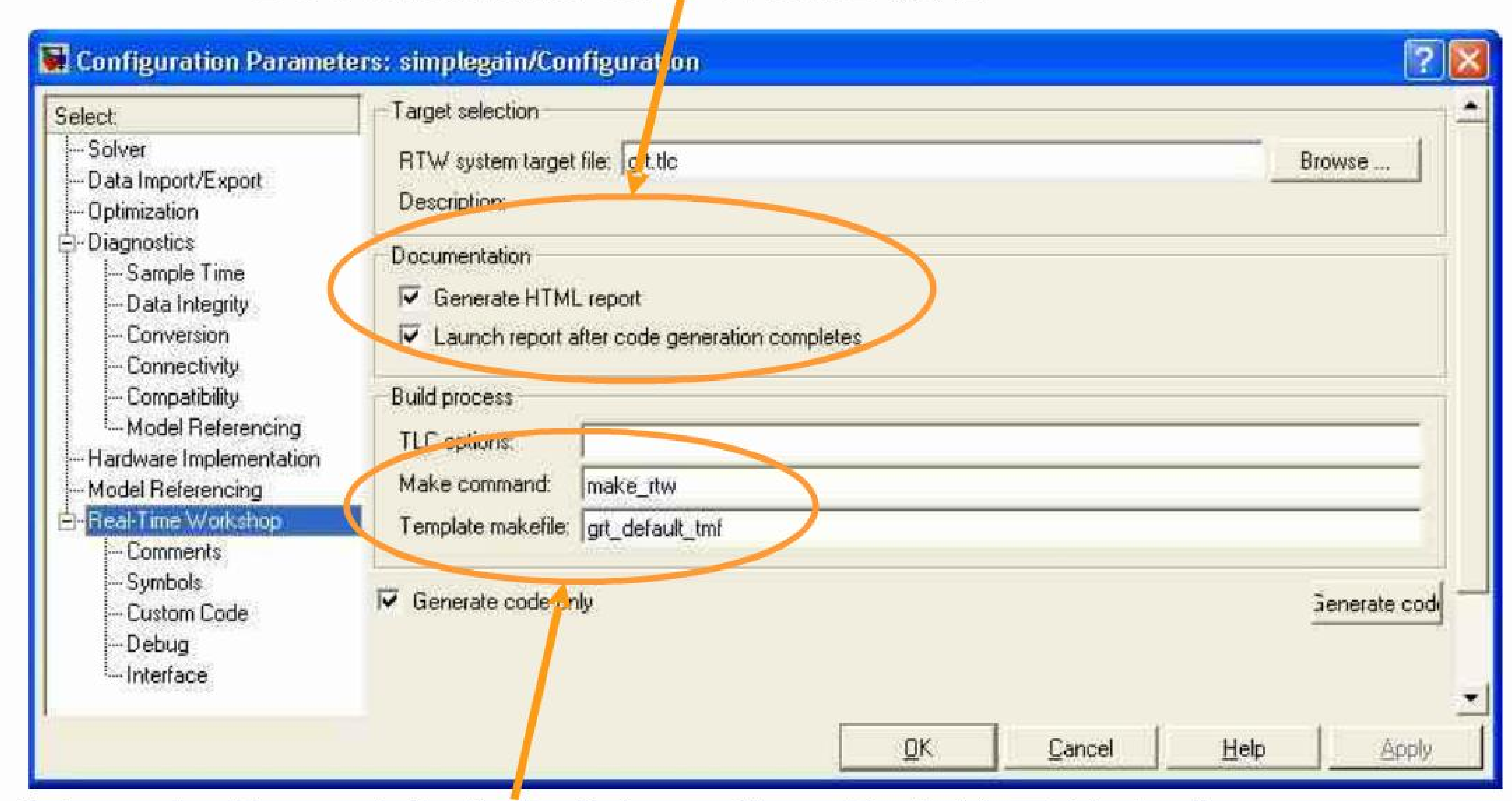
OK   Cancel   Help   Apply

```
>> simplegain
```



配置目标(续)

Documentation – HTML 报告



选择目标的同时也隐含选择了代码格式使用缺省选项



Real-Time Workshop 选项

通用(非目标特定)选项

Configuration Parameters: simplegain/Configuration

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Integrity
  - Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing
- Real-Time Workshop
  - Comments
  - Symbols
  - Custom Code
  - Debug
  - Interface

Simulation and code generation
- [x] Block reduction optimization
- [x] Conditional input branch execution
- [x] Implement logic signals as boolean data (vs. double).
- [x] Signal storage reuse
- [ ] Inline parameters
- Configure...
- Application lifespan (days): inf

Code generation
- [x] Enable local block outputs
- [ ] Reuse block outputs
- [ ] Ignore integer downcasts in folded expressions
- [x] Inline invariant signals
- [x] Eliminate superfluous temporary variables (Expression folding)
- Loop unrolling threshold: 5

目标特定选项 (GRT 目标设定截屏)其它目标可能还有另外的选项

OK | Cancel | Help | Apply



GRT 目标选项

Comments

- 全局控制
- 独立控制**Simulink**模块的注释，语句
  清除和存储类

Symbols

- 变量名的最大长度

Custom Code

- 引进定制的`.c`，`.h` 文件，初始化和结束函数
- 引进用户定制头文件，源文件和库文件的目录

Debug

- 控制编译输出
- 保留`.rtw` 文件
- **TLC** 调试

Interface

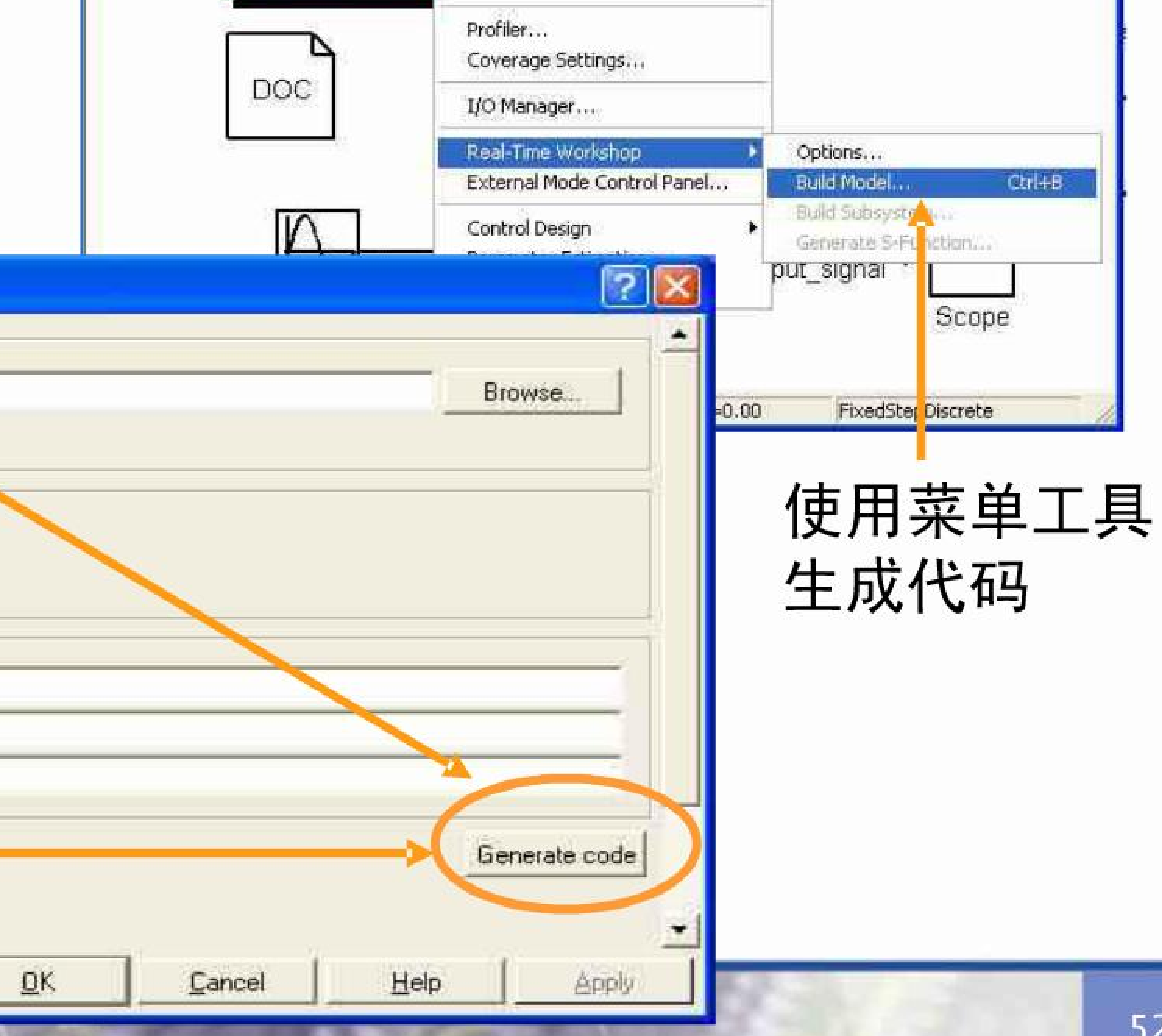
- **C** 代码的类型
- 控制数据交换模式
- **MAT-**文件数据记录



使用缺省选项生成代码

>> simplegain

使用参数配置对话框
生成代码



使用菜单工具
生成代码



Build 过程 – 做对比

```
>> simplegain
### Starting Real-Time Workshop build
    procedure for model: simplegain
### Generating code into build directory:
   c:class\coursefiles\RT01\simplegain_grt_rtw
### Invoking Target Language Compiler on
    simplegain.rtw
        tlc
        ...
### Loading TLC function libraries
### Initial pass through model to
    cache user defined code
### Caching model source code ...
### Writing header file simplegain_types.h
... (Writing other files)
### TLC code generation complete.
### Creating HTML report file
    simplegain_codegen_rpt.html
### Creating simplegain.mk from
   D:\Applications\matlab7lcs\rtw\c\grt\grt_vc.tmf
### Successful completion of Real-Time Workshop
    build procedure for model: simplegain
```

模块
& 目标
TLC 文件
gain.tlc,grt.tlc

目标语言
编译器

Make 模板
grt_lcc.tmf

Make

运行接口文件
targetname_main.c

停止
代码生成



生成的 C 代码文件概况

建立了名为 model_target_rtw 的目录，例如
simplegain_grt_rtw
生成的文件

Other Files

model.c
- 包括 model.h 和 model_private.h
- 代码的入口函数

model_data.c
- 一定条件下生成
- 模块参数
- 包括 model.h 和 model_private.h
- 常数

model_private.h
- 引入 Simulink 数据符号
- Real-Time Workshop - 特定细节 (各种宏，枚举，函数原型，代码内部的全局变量)

model.h
- 导出 Simulink 数据符号
- Model 数据结构，包括 rtM
- 模型入口函数



生成的 C 代码文件概况(续)

其他文件

model_types.h
- 模型结构前向声明 rtModel_model 和参数结构

rt_nonfinite.c
- 定义所有rt_nonfinite.中声明的非有限变量
- 需要时生成

rt_nonfinite.h
- 声明所有的非有限变量 rtInf，rtNaN 等等。

rtwtypes.h
- 包括 simstruc_types.h
- 具有更多为向后兼容作的头文件



模块结构

外部输入
(model_U)

模块 I/O
(model_B)

外部输出
(model_Y)

系统 / 模型

连续状态导数
(model_Xdot)

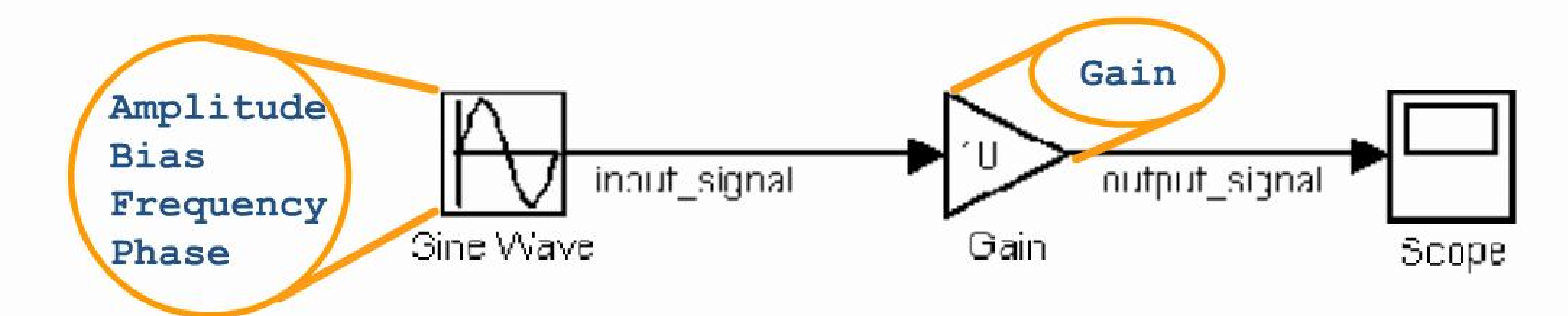
参数量
(model_P)

连续状态
(model_X)

工作向量
(model_DWork)



细看生成的代码



- 使用 HTML 报告观察文件 simplegain.c
- 模型如果没有外部输入/输出就不存在 model_U / model_Y 结构体
- 注意 GRT 接口函数名：MdlOutputs，MdlUpdate MdlInitializeSizes 等等。
- GRT 接口函数调用模型函数：simplegain_output，simplegain_update 等等。
- Sine Wave 模块和 Gain block 的参数如何在输出计算中被访问



参数结构

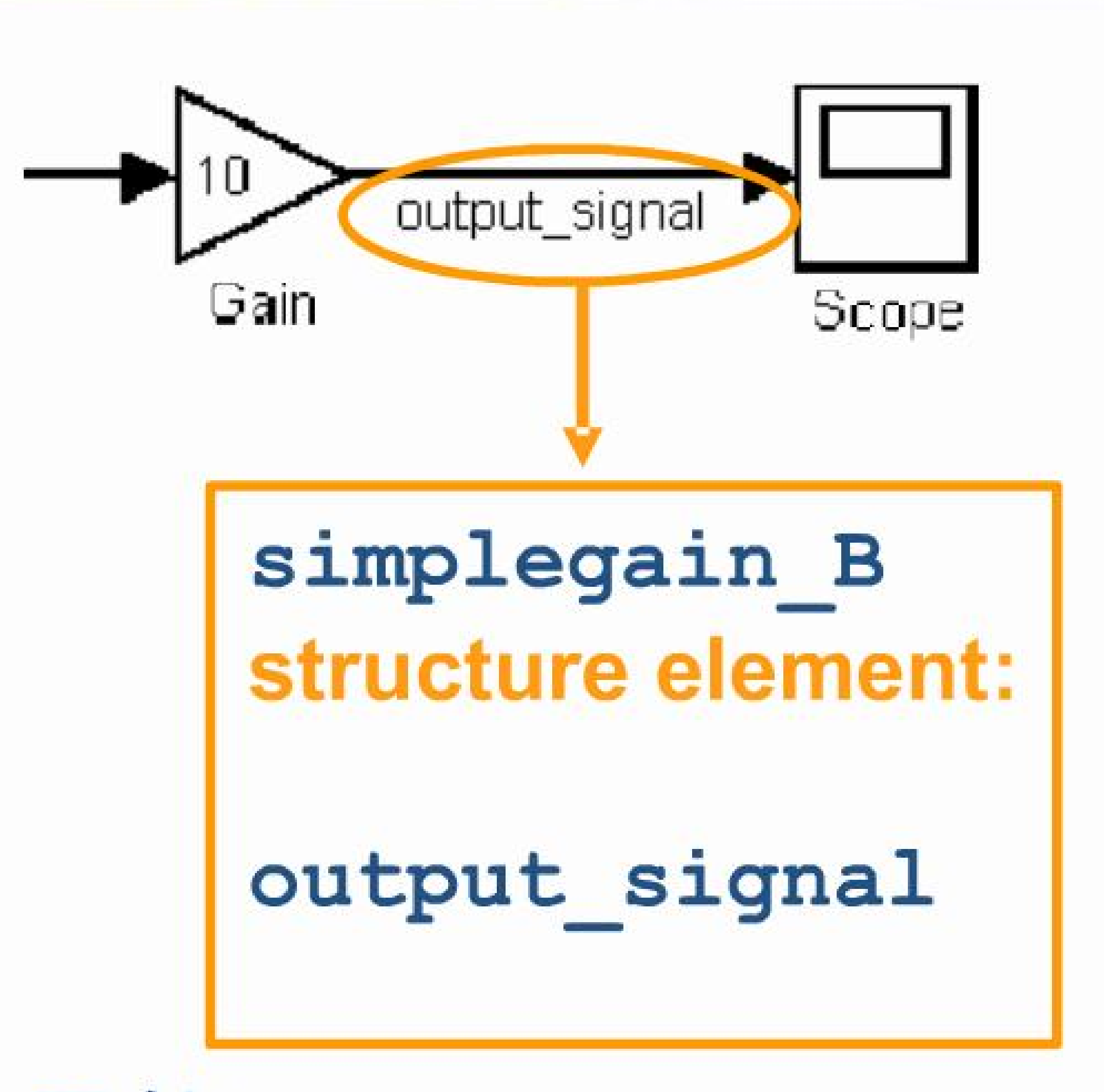
- 使用HTML报告观察文件 simplegain.h 和 simplegain_data.c
- 参数结构体定义在 simplegain.h 文件中
- simplegain_data.c 文件中创建了一个实例 simplegain_P
- 参数命名为 blockname_parametername
- simplegain_data.c 文件向结构体中的单元赋值
- 如果使用参数内嵌 simplegain_P 和 simplegain_data.c 文件就不会被创建。

simplegain_P **stucture elements are:**

```
SineWave_Amp
SineWave_Bias
SineWave_Freq
SineWave_Phase
Gain_Gain
```



模块 I/O 结构体



- 保存模块的输入和输出
- 在 simplegain.h 中定义并在 simplegain.c 中创建
- 储存在模块 I/O 结构体中的信号是全局的
- 某些仿真和代码生成优化选项会影响信号在结构体中的存储类型。



实时模型数据结构

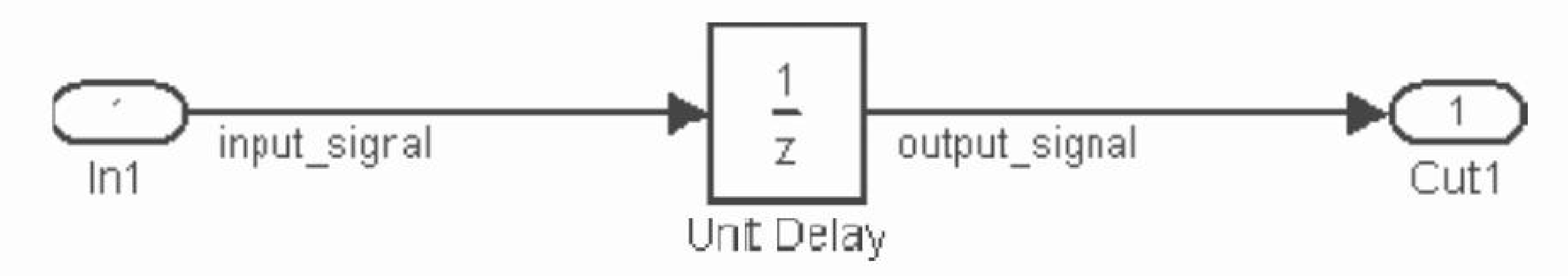
- 存储了根模型的信息
- 定义为 rtModel_model(rtModel_simplegain)
- 指向实例的指针，称为 model_M(simplegain_M)
- 保存指针指向:
  - 模型路径和模型名字
  - 指向所有子S-functions 指针的列表
  - 出错状态
  - 求解器
  - 定时
  - 数据记录等等。

与 ERT 比较

- rtModel_model 被简化以节约存储空间
- 可以彻底摒弃这个结构体



示例：生成离散模型的代码



- 从示例目录打开模型 singledisc.mdl。
- 模型包含一个离散状态。
- 模型配置为使用：
  - 离散定步长求解器，步长为auto
  - GRT 目标
  - 通过 input port 模块引入外部信号
  - 通过 output port 模块返回时间，状态和输出值
  - unit delay 模块采样时间 0.1 秒，初始值为 -1.0
  - 只生成代码

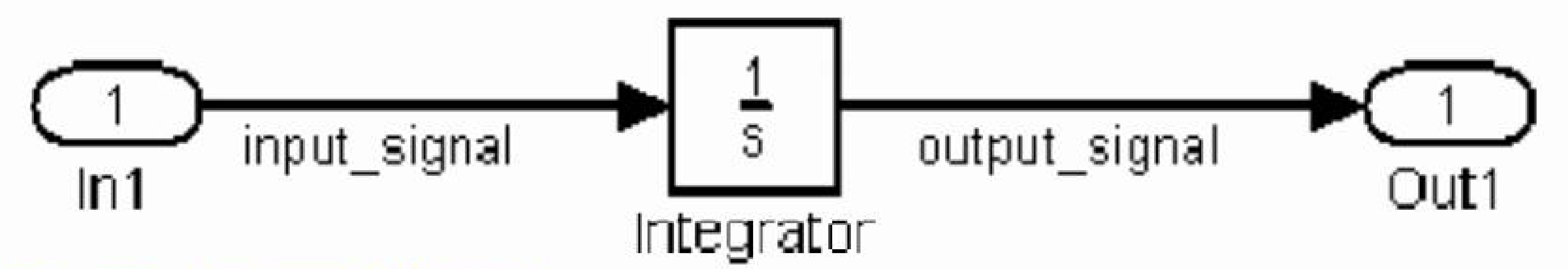


示例：生成离散模型的代码(续)

- 生成代码并用HTML报告查看代码。
- 注意源文件singledisc.c中以下信息
  - 从外部输入使用singledisc_U处理。
  - 向外部输出使用singledisc_Y处理。
  - Unit delay 模块的初始条件保存在 singledisc_P中(在singledisc_data.c中初始化)。
  - 离散动态在singledisc_update 函数中处理。
  - 离散状态保存在结构体域UnitDelay_DSTATE 的 singledisc_DWork 中。
- 注意：名字中的空格（模块名和信号名）在Real-Time Workshop代码生成时被剔除。



示例：生成连续模型的代码



- 从示例目录打开模型 `singlecont.mdl`
- 模型包含一个连续状态
- 模型配置为：
  - 使用连续定步长求解器，步长值为 auto
  - 生成 GRT 代码
  - 通过input port 模块引入外部信号
  - 通过output port 模块返回时间，状态和输出值
  - 设定积分器的内部初始状态为 -1.0
  - 只生成代码(不build目标)



示例：生成连续模型的代码(续)

- 生成代码并用HTML报告查看代码。
- 注意源文件 singlecont.c 中的以下信息：
  - 从外部输入使用 singlecont_U 处理。
  - 向外部输出使用 singlecont_Y 处理。
  - 积分器的初始条件保存在 singlecont_P 中 (在 singlecont_data.c 中初始化)
  - 导数方程在函数 singlecont_derivatives 中实现
  - 在 singlecont_derivatives 中
    - 导数来自于模型结构体 singlecont_M
    - 保存在字段 Integrator_CSTATE 中，结构指针 singlecont_Xdot指向它
    - 重新赋予新的值
  - 积分算法被调用以积分出连续状态量。



配置模型用于ERT目标

配置目标

需要 Real-Time Workshop Embedded Coder


```
>> singledisc
```



ERT 选项

■ HTML 报告中包括代码到模型选项的超链接

Comments

■ 同 GRT 中一样

■ 定制的注释有：

- Simulink 模块描述
- Simulink 数据对象描述

Custom Code

■ 包括定制的.c，.h文件，初始化和结束函数

■ 定制头文件源文件和库文件的目录

Debug

• 控制编译输出

• 保留.rtw文件

• TLC 调试

Symbols

■ 同 GRT 中一样

■ 最小变量长度

■ Simulink 数据对象命名规则



ERT 选项(续)

Interface

- 同 GRT 中一样
- 可以选择的支持项：
  - 浮点数支持
  - 绝对时间
  - 连续模块
  - 非有限数值
  - 非内嵌 S-functions
- 使用一个函数计算输出和状态更新
- 兼容 GRT 的调用接口。
- 实时模型结构中可选的误差状态选项。
- 可选的 MAT-文件数据记录。

Templates

- 可选主程序样例
- 可选择针对用户目标环境的定制模板

Data Placement

- 指向用户存储类对象声明和定义



使用缺省选项生成 ERT 代码

Configuration Parameters: singledisc/Configuration

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Integrity
  - Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing
- Real-Time Workshop
  - Comments
  - Symbols
  - Custom Code
  - Debug
  - Interface
  - Templates
  - Data Placement

Target selection
RTW system target file: ert.tlc [Browse...]
Description: RTW Embedded Coder (no auto configuration)

Documentation
☑ Generate HTML report
☑ Include hyperlinks to model
☑ Launch report after code generation completes

Build process
TLC options:
Make command: make_rtw
Template makefile: ert_default_tmf

Custom storage class
☐ Ignore custom storage classes

☑ Generate code only [Generate code]

[OK] [Cancel] [Help] [Apply]

- 创建的文件如同 GRT
- 创建了ert_main.c(样例主程序)
- 如果为选择生成样例主程序则创建autobuild.h



分析生成的 ERT 代码

- 在 HTML报告中查看生成的文件
- 注意下列信息：
  - 在文件 singledisc.c 中
    - 模型的输出和更新计算收到一个名为singledisc_step 的函数中
    - 所有模块的结构体的处理方式与 GRT 中类似
  - 在文件 singledisc.h 中
    - 实时模型数据结构体精简为 char *
    - 可以通过选项清理模型数据结构体中的错误状态信息



ERT 的 main() 函数

- 实现仿真循环的主程序行
- 该程序行保存在 ert_main.c 中
- 如果该选项选中将生成样例main程序
- 主程序实现了下述内容：
  - 初始化模型
  - 安装中断服务程序(ISR)
  - 执行后台任务
  - 进行 cleanup 操作



rt_OneStep

- ert_main.c 包括 rt_OneStep 的三种实现，取决于模型中采样速率的个数和求解器的模式：
  - 单任务(Single tasking)
  - 多速率多任务(Multi-rate multi tasking)
  - 多速率单任务(Multi-rate single tasking)
- 在多速率模型中
  - 每个运行在给定采样速率的模块赋予任务标识task identifier (tid)
  - 任务具有优先级，按速率降序排列
  - 每次引用，rt_OneStep 都会多次调用 model_step，传递给合适的tid。

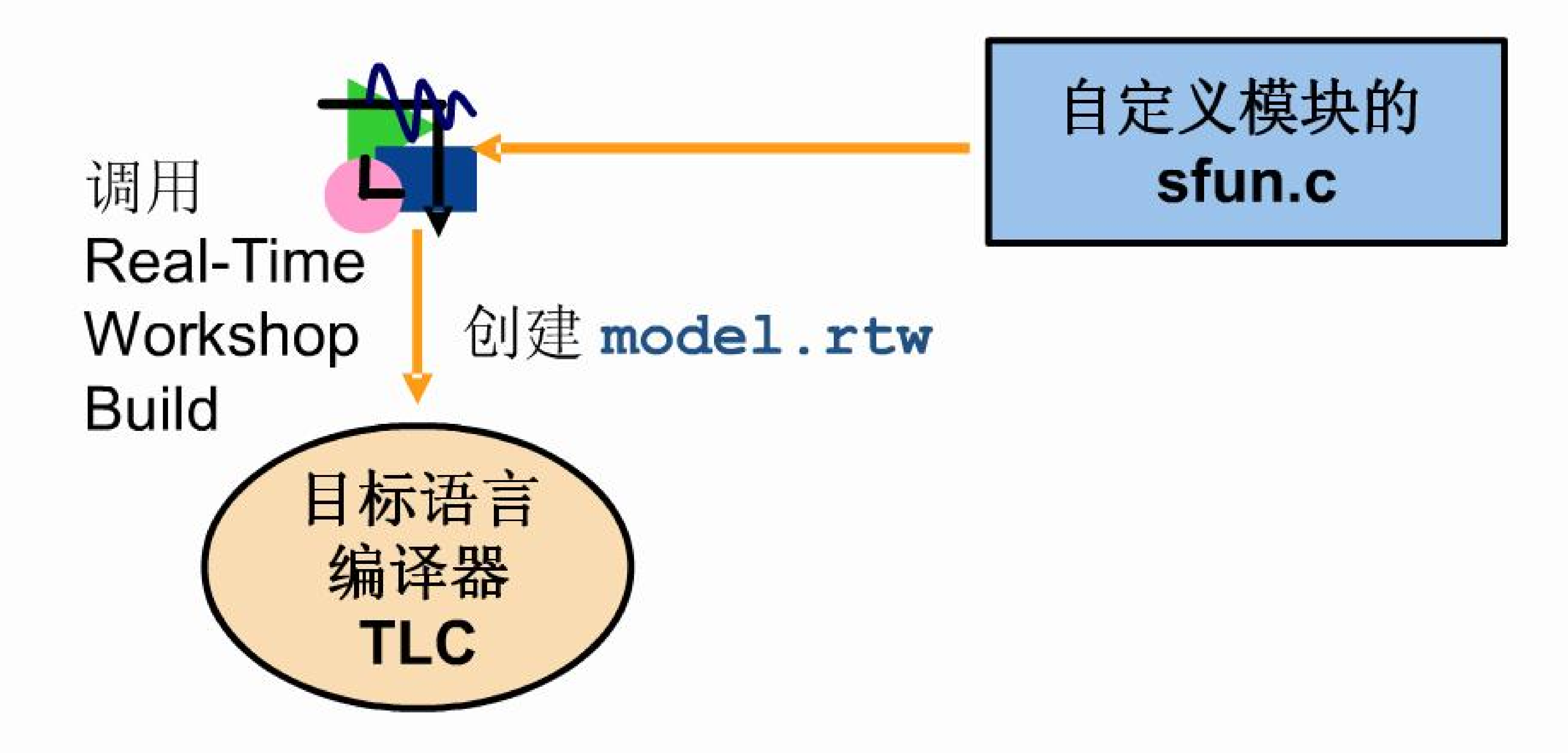


配置模型并生成 **ERT** 代码



生成代码的步骤

第一步生成 **model.rtw**



- model.mdl 编译为记录文件model.rtw
- Target Language Compiler 解析该文件，生成 C 代码



代码生成各阶段(续)

TLC 解析 model.rtw 文件并生成目标专属的C文件

两种类型TLC文件：

- 模块TLC文件(生成每个模块的代码)
- 目标TLC文件(规定代码结构)



代码生成各阶段(续)

生成Make文件并进行
make 过程

model.c, model.h
...

Make模板
grt_lcc.tmf

标记扩展

model.mk

Make

运行接口文件
targetname_main.c

model.exe

目标

生成的代码可以：

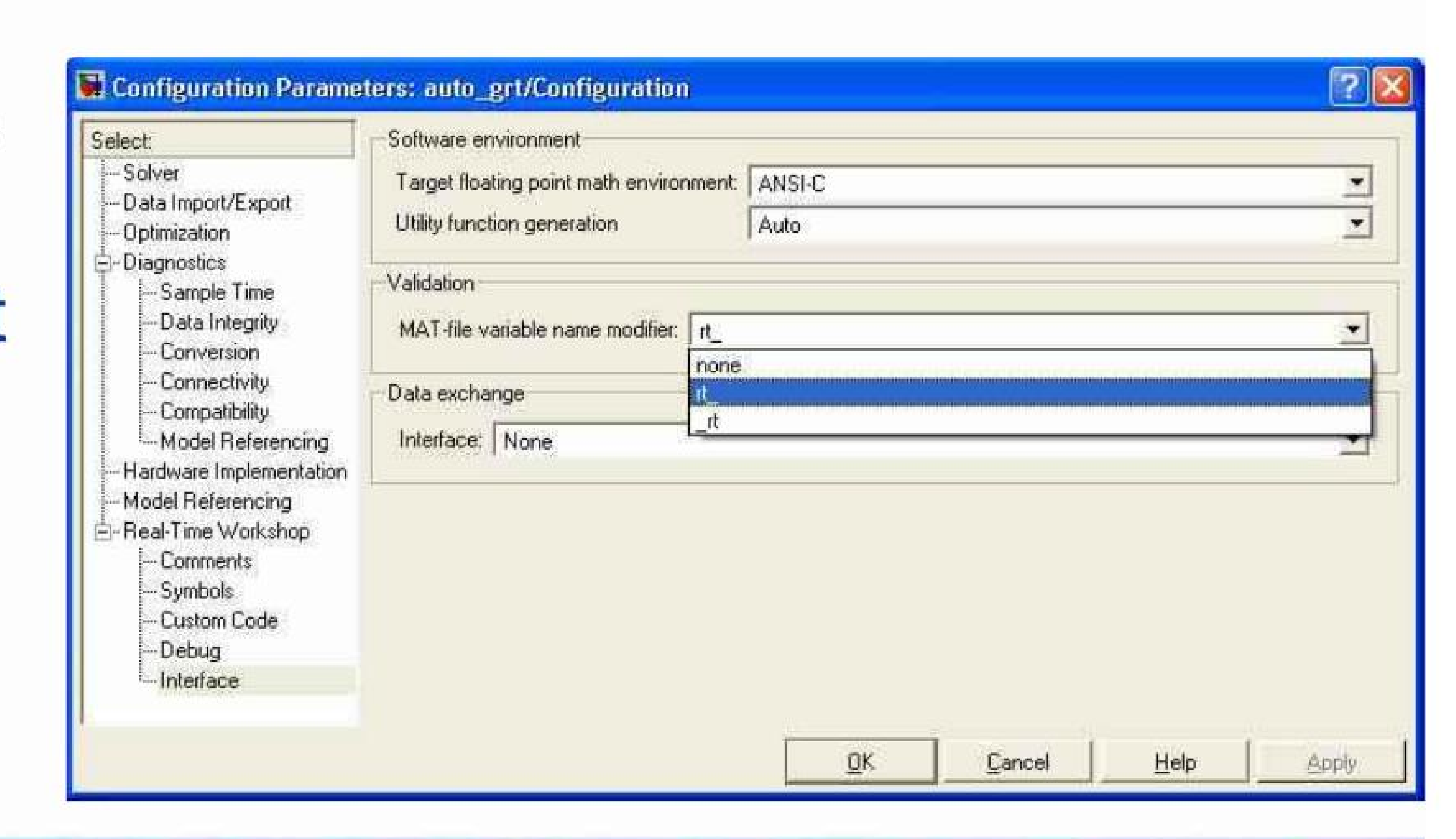
- 在主机平台上编译使用
- 不编译
- 交叉编译

- 模板 make 文件 (target_compiler.tmf) 通过标记扩展变为model.mk
- 需要修改target.tmf以适应定制运行环境



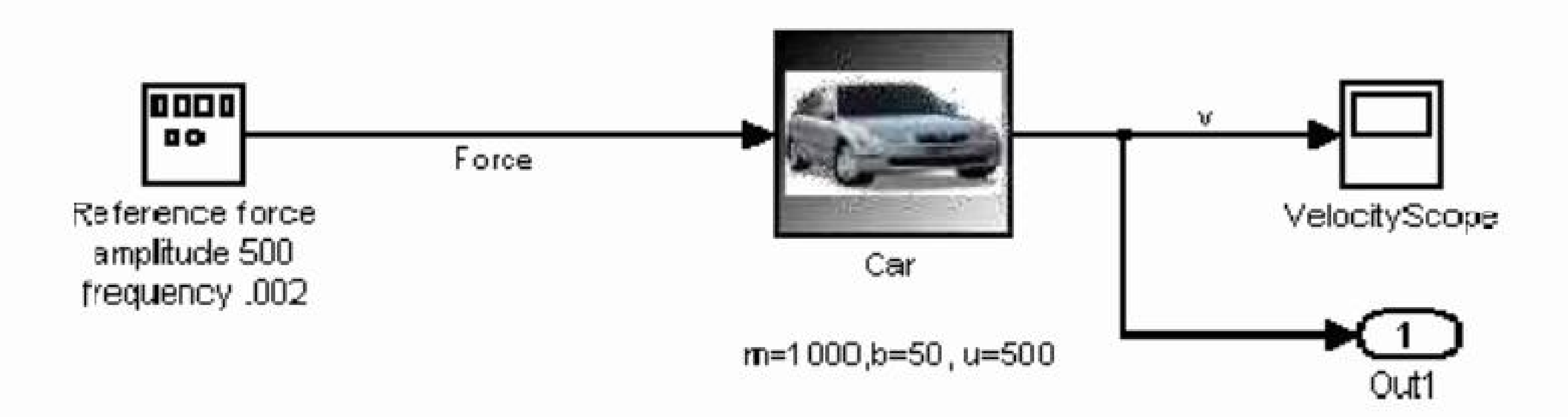
记录目标的数据

- 各种目标都支持信号/数据记录，包括GRT和ERT
- 模型必须配置为使用下列模块之一：
  - Scope 模块
  - To Workspace模块
  - To File模块
  - 顶层的 Outport模块
- 数据记录到MAT-文件model.mat
- 可添加前缀，后缀来修改变量名(rt_ 或 _rt)





示例：运行生成代码和代码检查



- 打开示例目录下的 auto_grt.mdl 模型
- 配置将信号 v 输出到MATLAB
  - 添加一个output port 模块
  - 通过选择Configuration Parameters，Workspace I/O 的选项将时间，状态和输出写到工作区(workspace)
- 运行模型
- 检查工作区中的 tout，xout和yout变量



示例：运行代码生成并检查代码(续)

- 配置模型为生成GRT代码
- 在 Configuration Parameters 对话框中：
  - Real-Time Workshop 选项下：
    - 选择生成GRT代码
    - 取消Generate code only
  - 在 Real-Time Workshop，Interface 选项中
    - 选择MAT-文件修饰符为rt_
- 保存并编译模型
- 运行模型
- 检查仿真结果



示例：运行代码生成并检查代码(续)

Figure 1

File Edit View Insert Tools Desktop Window Help

GRT Target executable results Vs Simulink simulation results

Simulink Simulation Results
GRT Target Executable Results

Velocity

Time

Figure 1

File Edit View Insert Tools Desktop Window Help

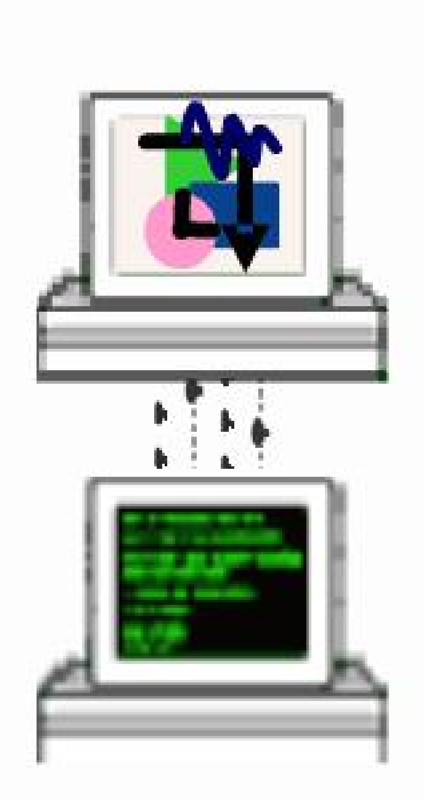
Difference between Simulink simulation and standalone executable

Time

生成代码并检查



小结



- 实时程序的构架
- 代码格式
- GRT target 的目的
- 生成简单系统 GRT 目标的代码
  - GRT 代码生成选项
- Build 的过程 – 对比
- 生成的文件和代码结构概况
- 模块的结构
- 实时数据的结构
- 示例：生成离散和连续系统模型的代码
- 配置并生成 ERT 目标的代码
  - ERT 目标的代码生成选项
- 代码生成的过程(阶段)
- 信号记录



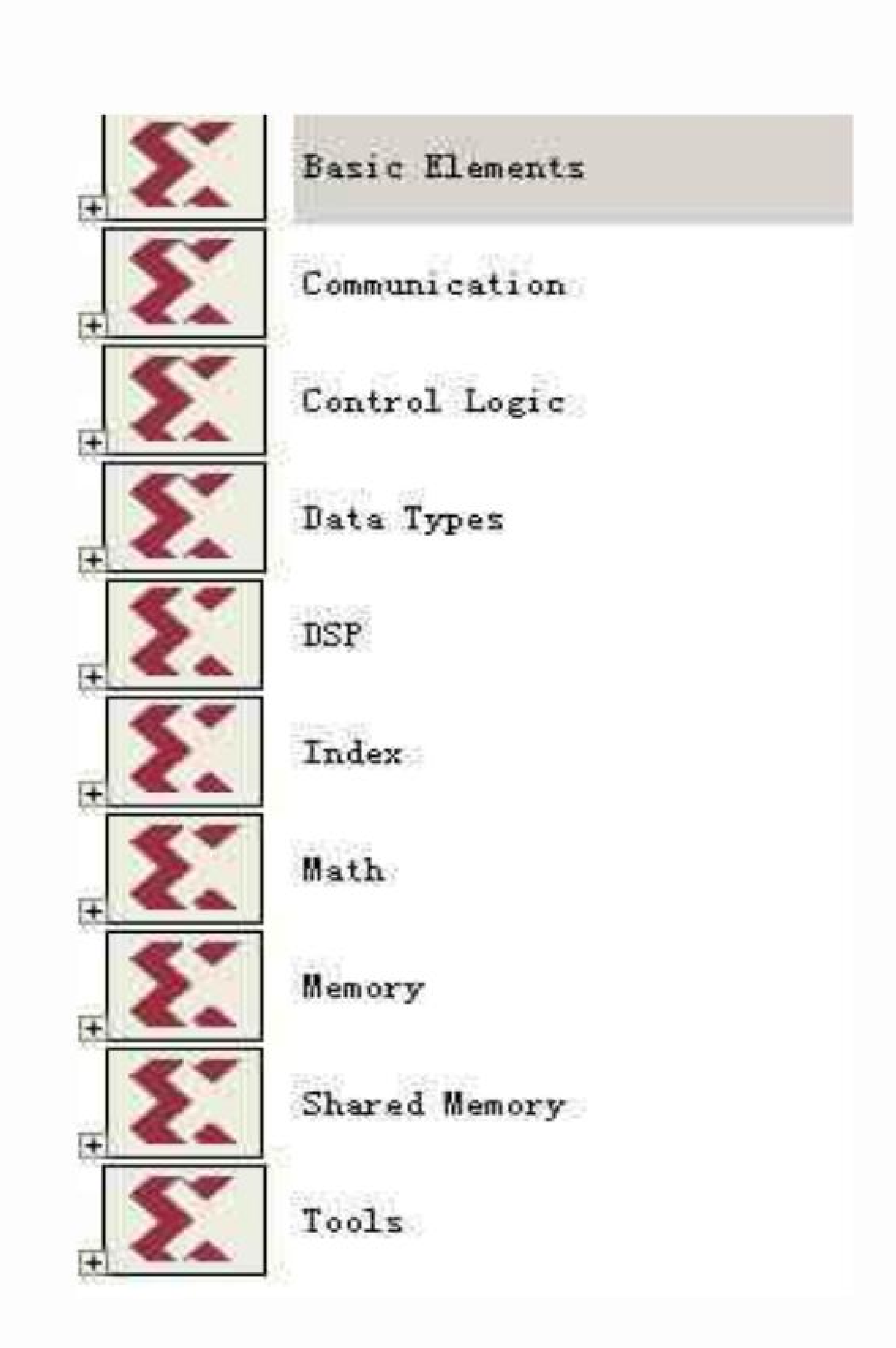
课程概要

- Simulink建模仿真
- 面向DSP的自动代码生成
- 面向FPGA的自动代码生成



概要

- 基本特性
- 安装与使用
- 使用基础
- 自动代码生成
- 时钟控制
- 将生成结果纳入设计





System Generator 的基本特性

- 支持VHDL和Verilog的自动代码生成
- 支持软硬件协同仿真
- 支持网络方式和以太网方式
- 支持Xilinx的开发工具
  - 与EDK交互
  - 对硬件自动综合和软件接口的抽象
  - 自动生成对外设和协处理器的抽象
  - 生成RTL
  - 对Simulink集成
  - 使用黑盒子插入VHDL和Verilog



System Generator 的安装与下载

- 软件要求
  - MATLAB v7.0.1/Simulink v6.1 (R14.1) Service Pack 1, MATLAB v7.0.4/Simulink v6.2 (R14.2)Service Pack 2, or MATLAB v7.1.0/Simulink v6.3 (R14.3),
  - Xilinx ISE Foundation or Alliance, release version v8.1.01 (8.1 Service Pack 1) or later.
- 软硬件安装
  - 硬件方面: 安装好FPGA开发板
  - 软件方面: 安装好Systemgen后,可以再下载新的plugin来支持更多的板卡,安装方法
    - 下载相应的plugin
    - 在matlab中,把当前目录设置到plugin的位置
    - 在matlab里面输入 `xlInstallPlugin('myplugin.zip')`



使用基础

■ 一个简单的例子

▶ $Y(n+1) = x(n+1) + a\,x(n-1)$　　相当于一个简单的低通滤波器

tutorial_simplefir *

File Edit View Simulation Format Tools Help

200 Normal

System Generator

Sine Wave

Sine Wave1

double

Gateway In

Fix_14_12

Delay

Constant

Fix_2_0

Mult

Fix_16_12

AddSub

Fix_17_12

Gateway Out

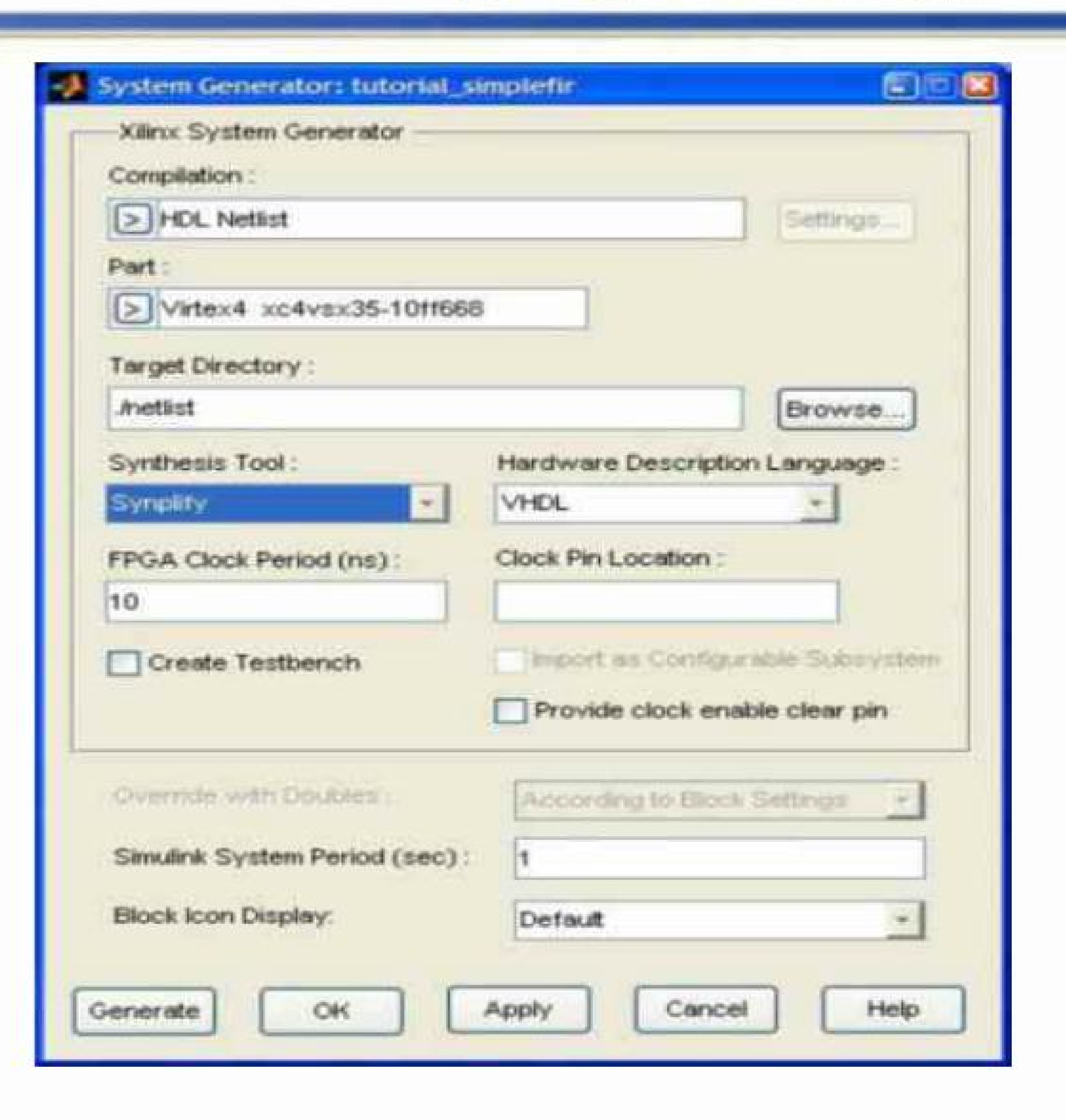
Scope

Ready 94% ode45



具体步骤

- 新建一个模型
- 选择需要的模块,并拖拽到模型里面
- 定义具体的参数
- 连接各个模块

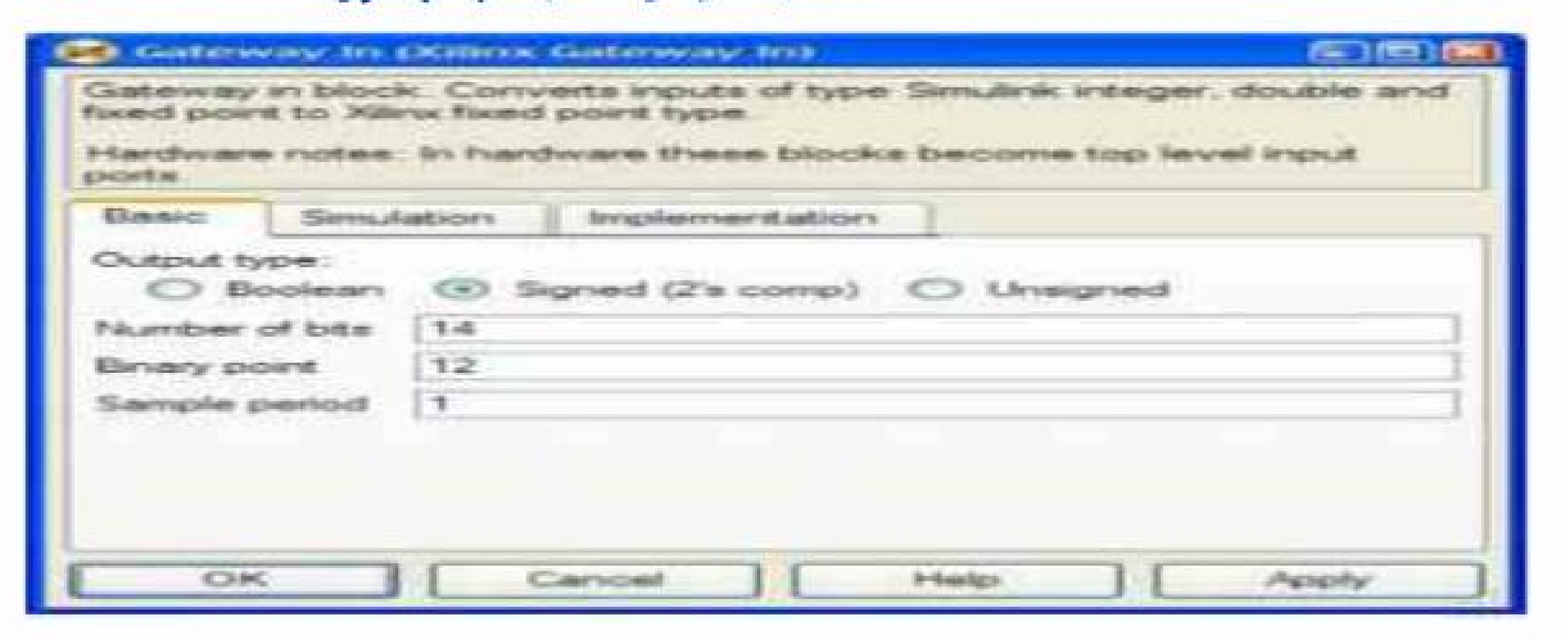


System generator模块

- compilation参数
- part参数
- 综合工具
- 硬件描述语言

System generator模块和Simulink模块的交互

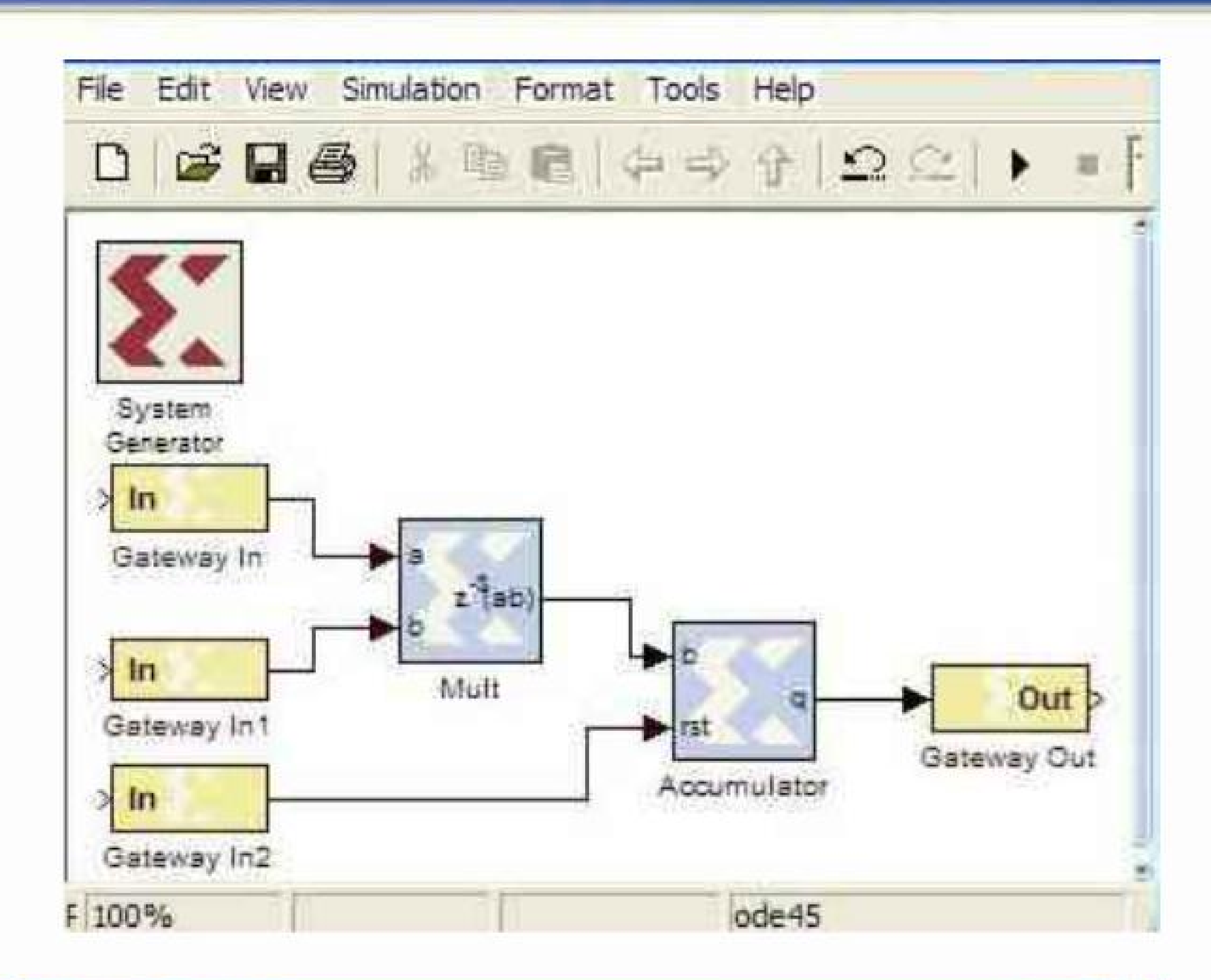
- 相互的关系
- gateway模块
- 两种时钟的概念

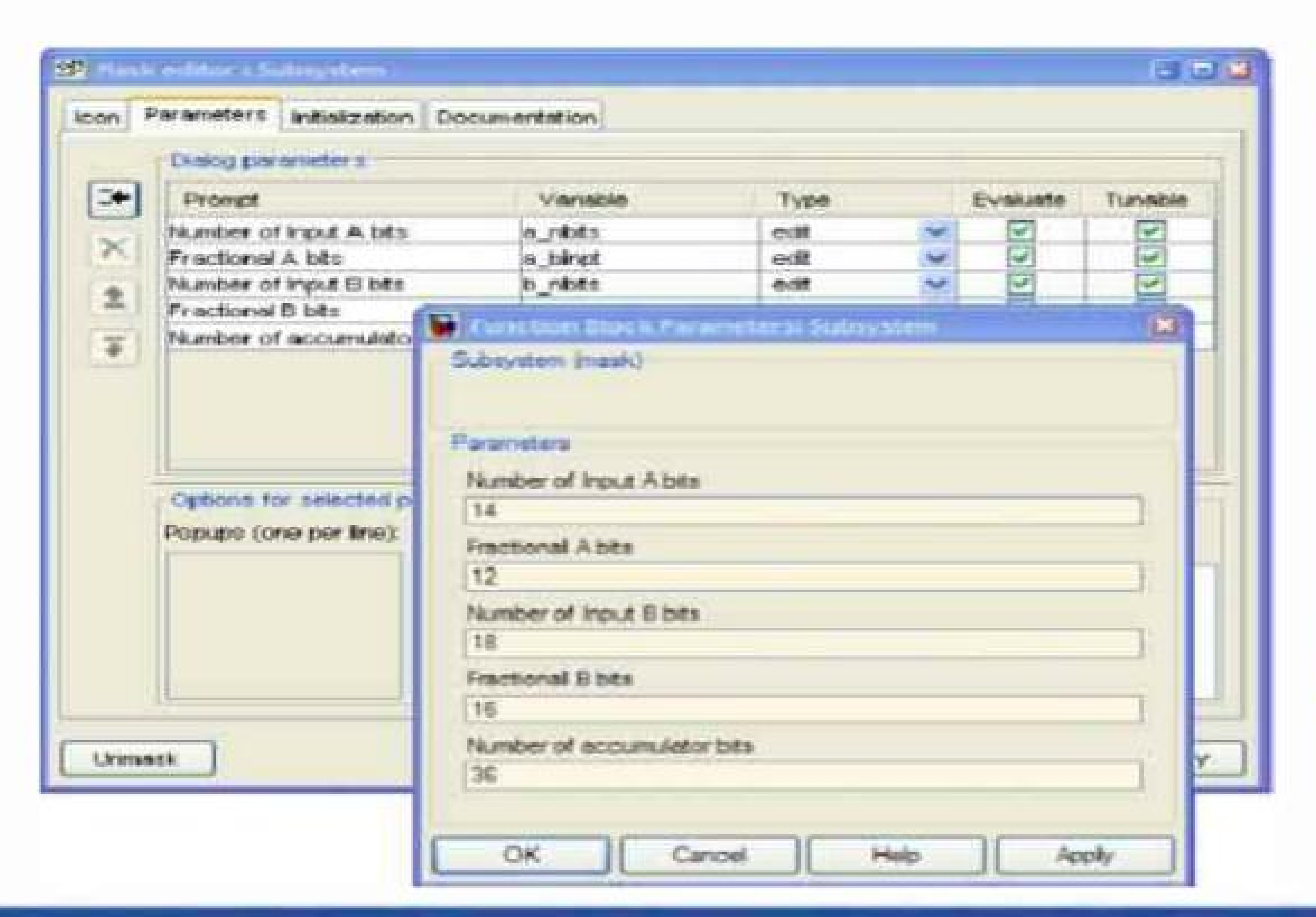


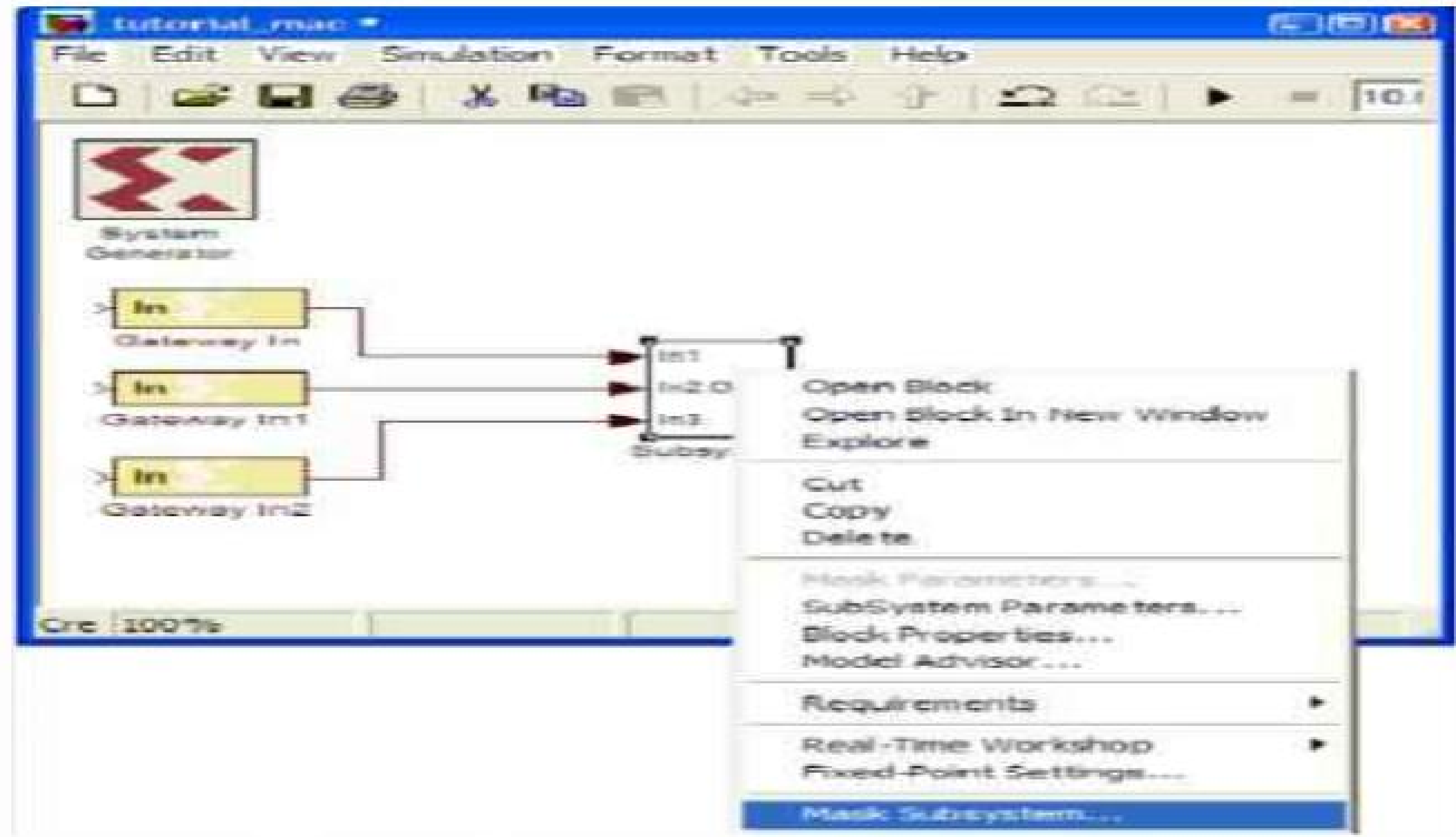


子系统与MASK参数

- 子系统的意义
- 子系统的建立
- MASK参数的意义
- MASK参数的使用
- 从Workshop来MASK参数

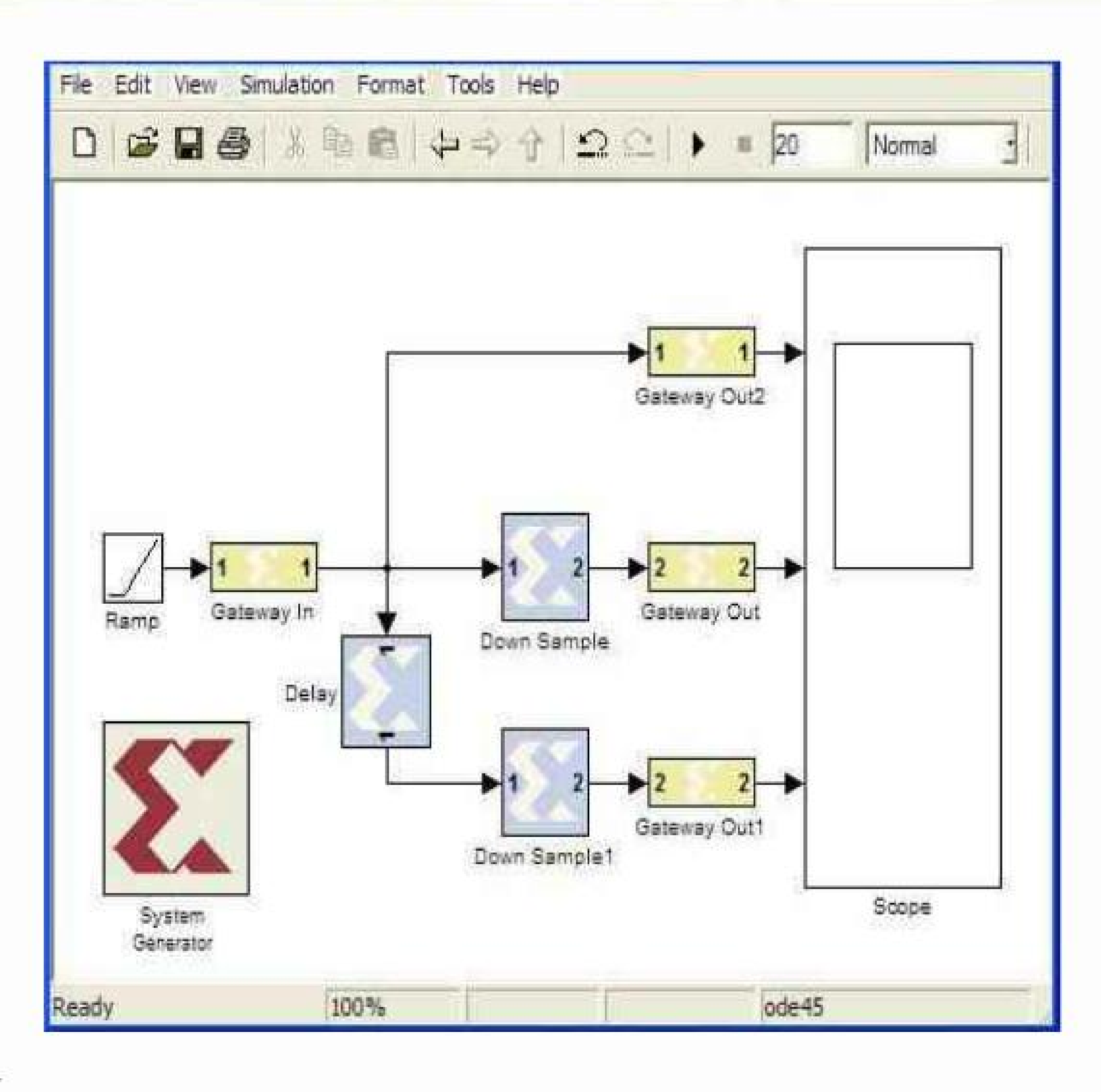








采样率变换模块

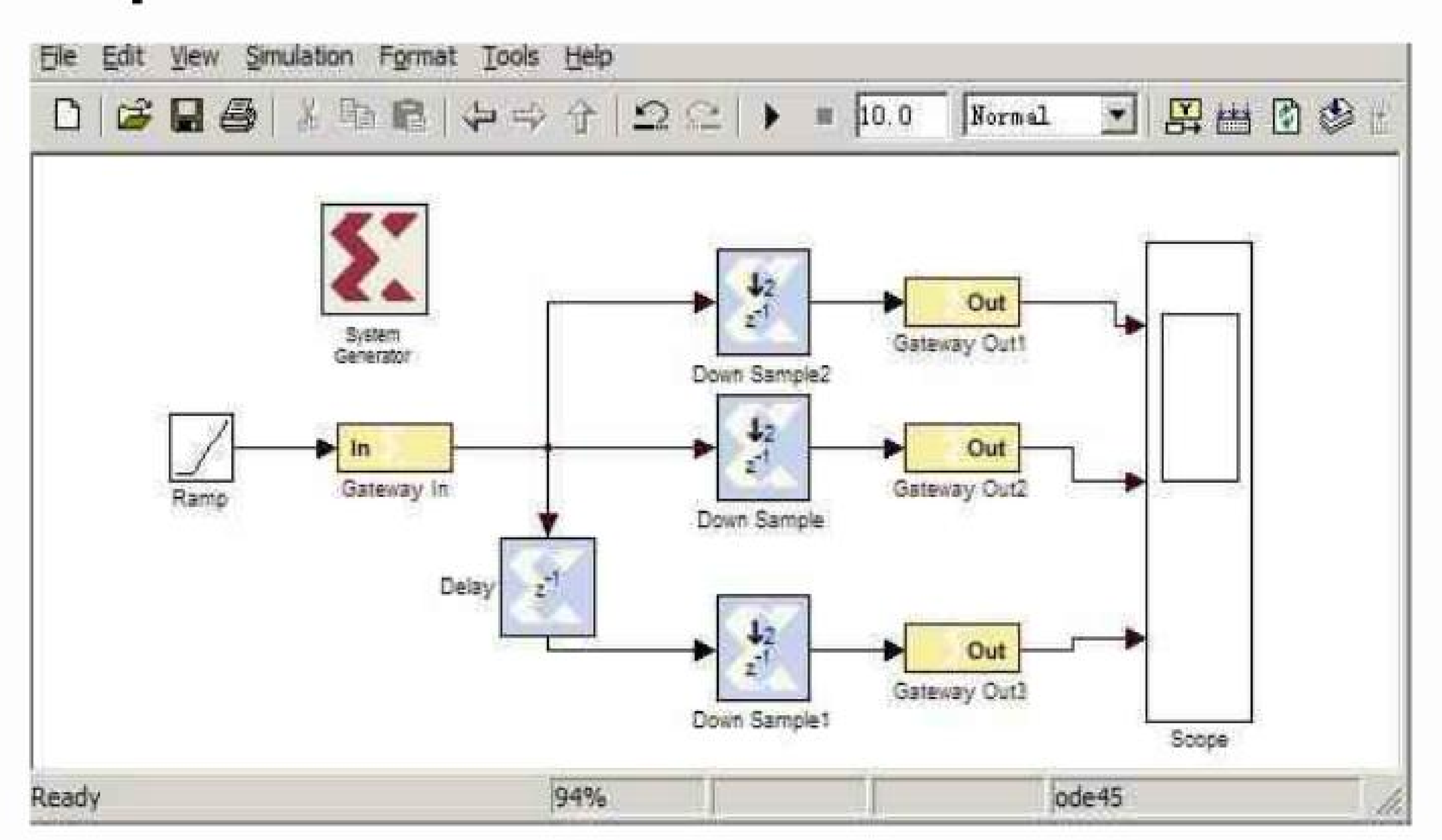


- Up Sample, Down Sample
- Assert
- Time Division Multiplexer,
  Time Division Demultiplexter
- Serial to Parallel , Parallel to Serial
- Dual-part memory, Register, Addressable Shift Register
- FIR filter(多相位）

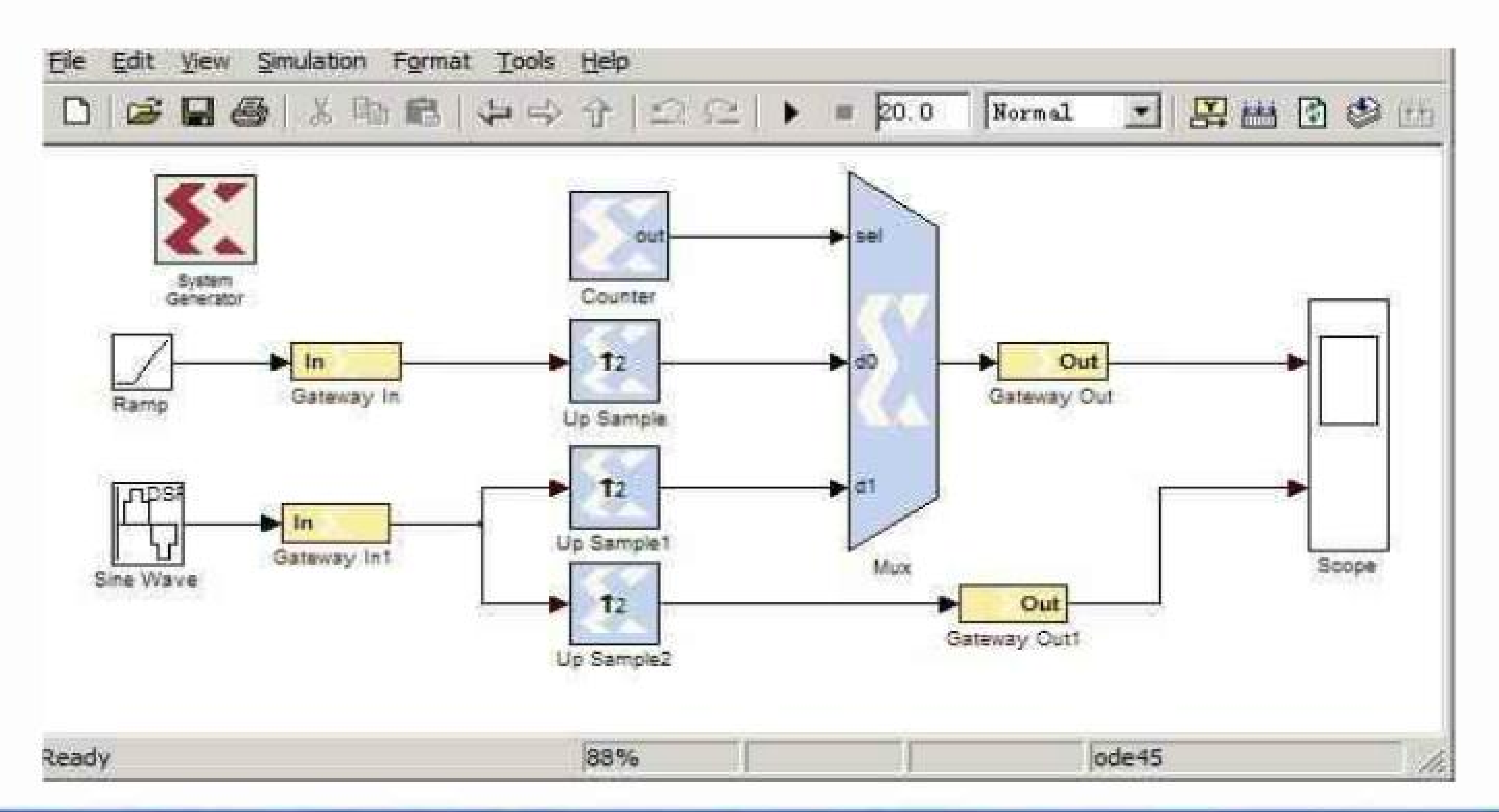


Up Sample和Down Sample

- Down Sample
  - 用途
  - 参数说明
  - 两类方式



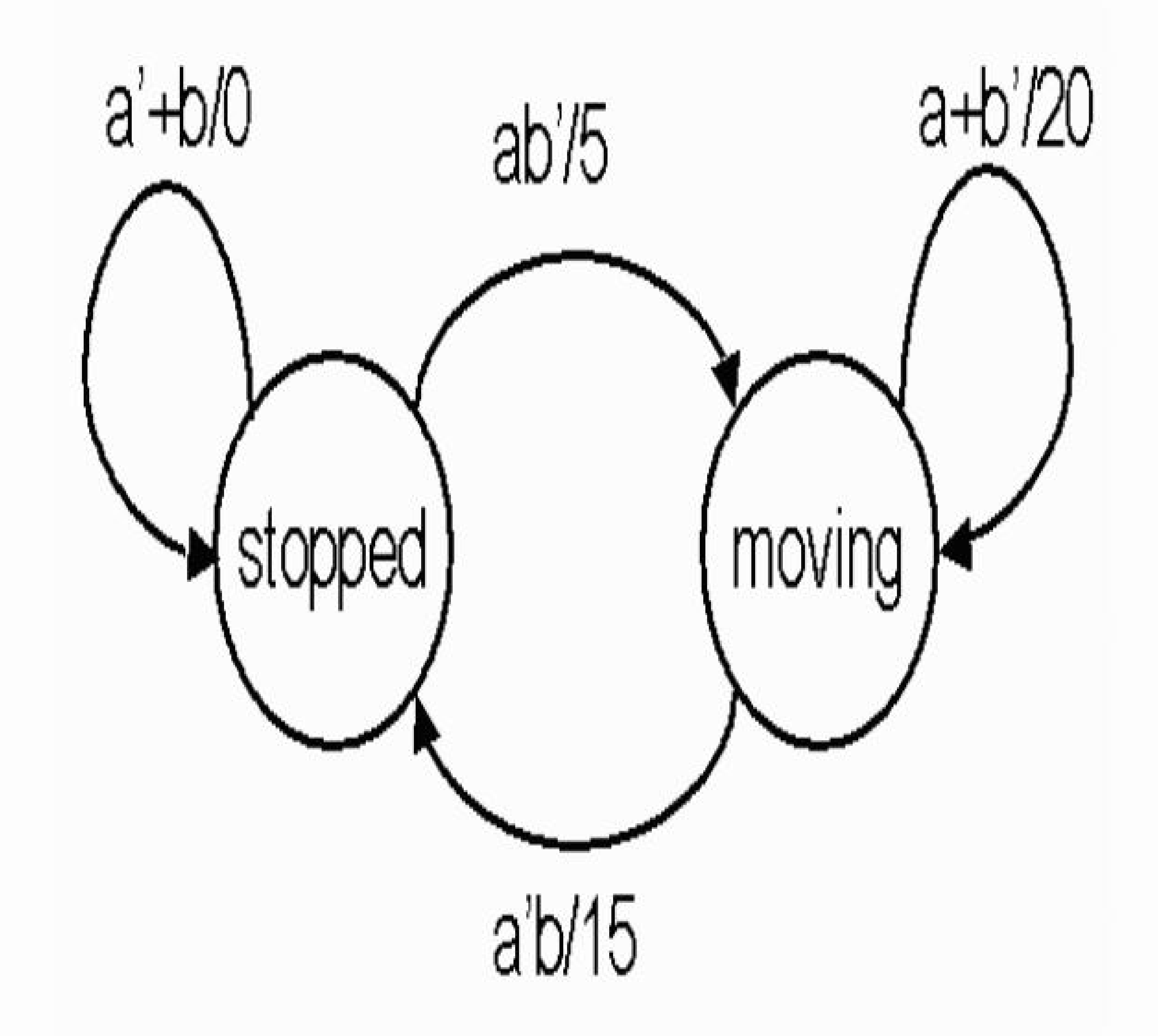
- Up Sample
  - 用途
  - 参数说明
  - 两类方式





分支结构

- 对于利于用语言来描述的分支结构，例如IF, Else ,Switch,system generator 提供了几种处理机制
  - 黑盒子的方式插入HDL
  - Mcode插入M-Function
  - 8-bit的微控制器

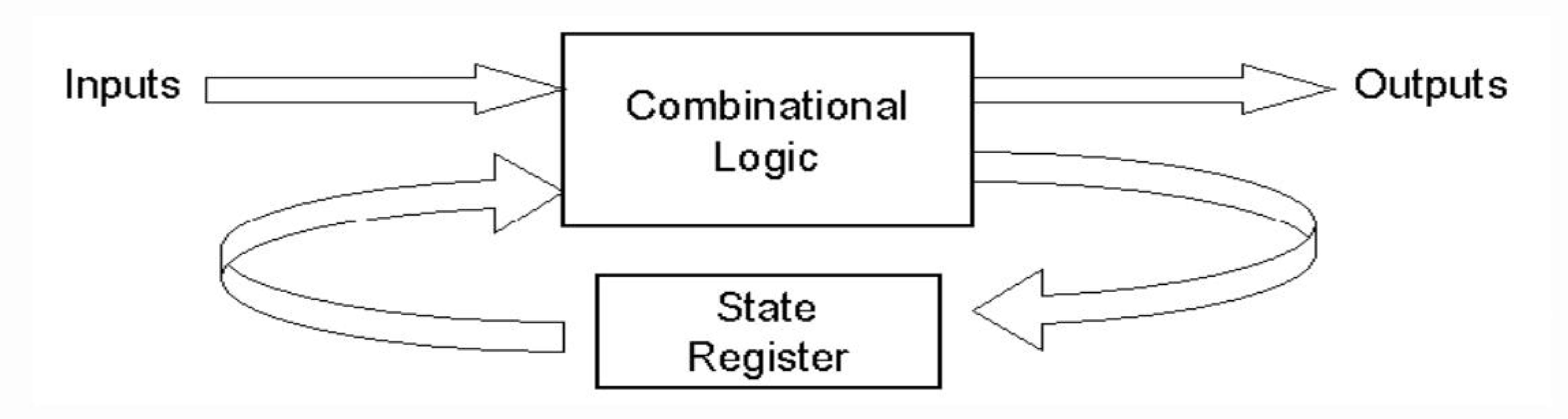




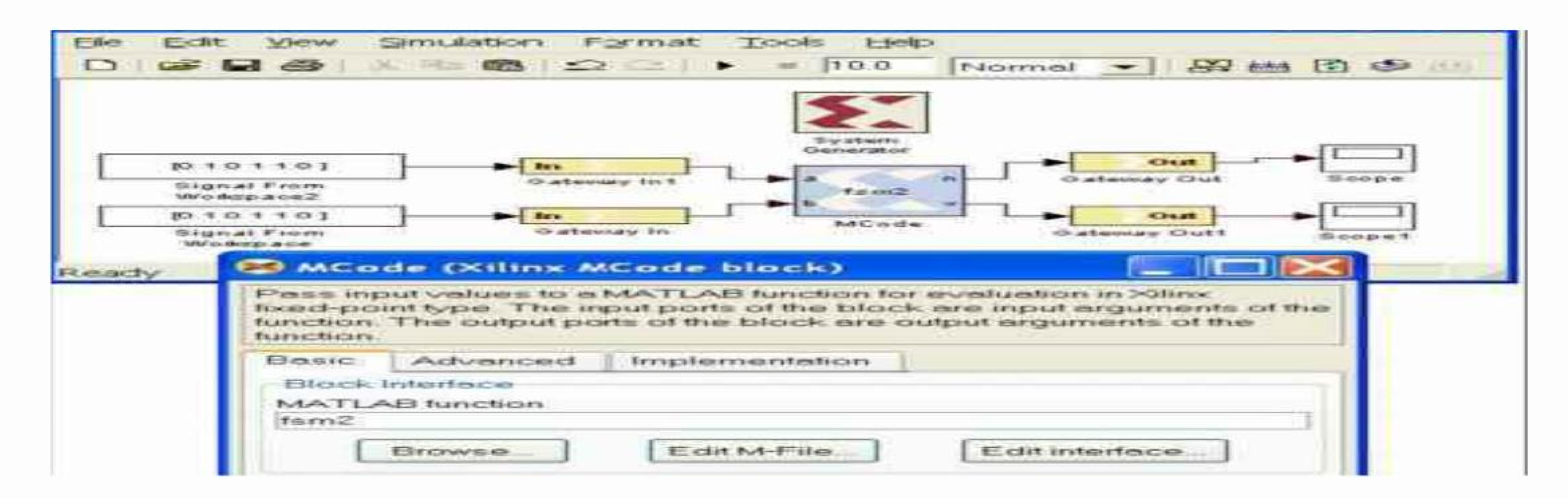

```
function [n,v] = fsm(a,b)
 stop = 0, moving = 1; % symbolic constants
 proto = {xlUnsigned,1,0};
 persistent s, s= xl_state(stop,proto);
  switch s;
      case stop
                  if a & ~b, n = 1, v = 5;
                  else, n = 0, v = 0;
                  end
      case moving
                  if a | ~b, n = 1, v = 20;
                  else, n = 0, v = 15;
                  end
      otherwise, n = 0, v = 0;
      s = n;
 end
```



System Generator的实现原理

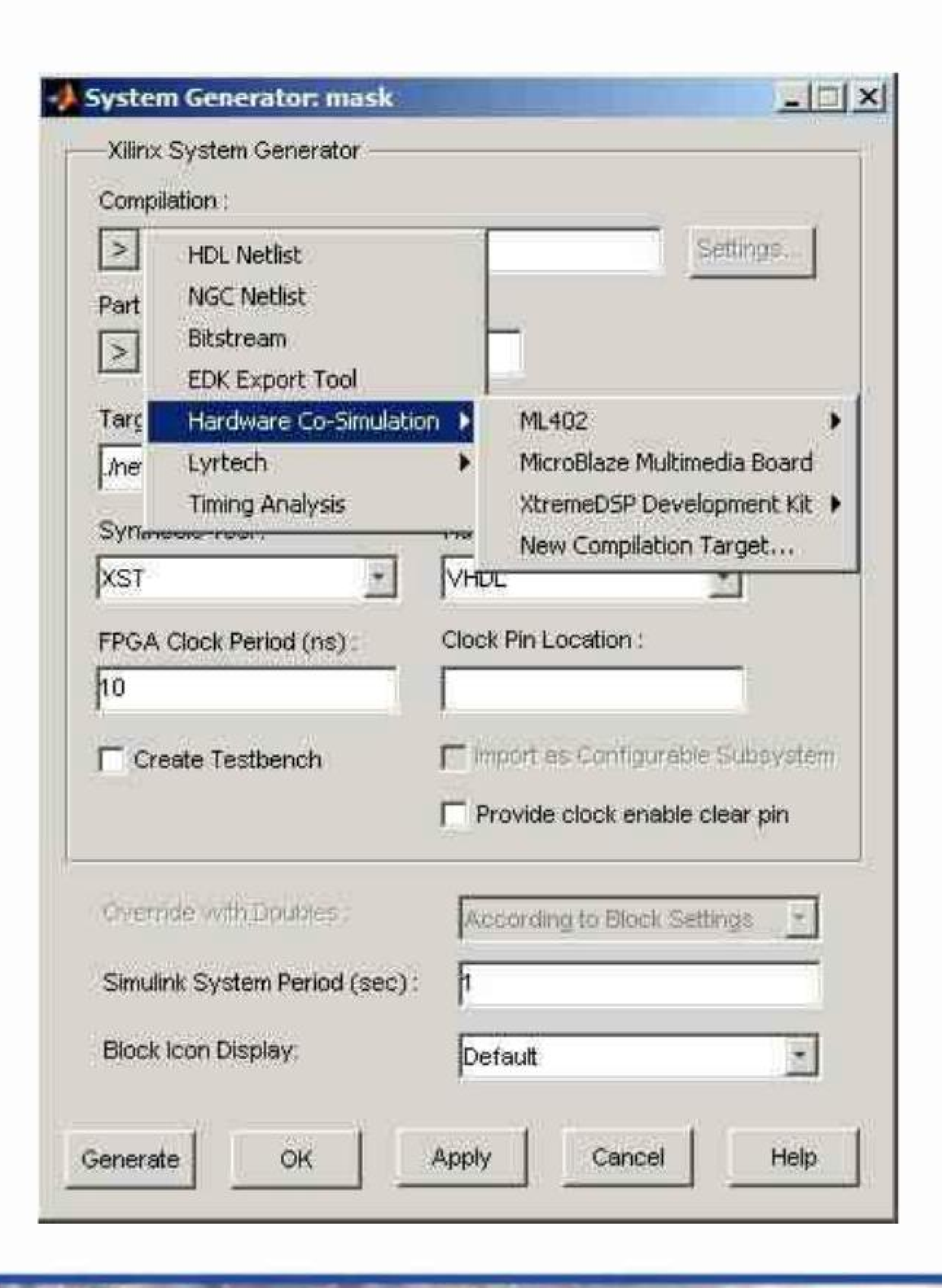


Mcode的实现：




恒润科技
HIRAIN TECHNOLOGIES

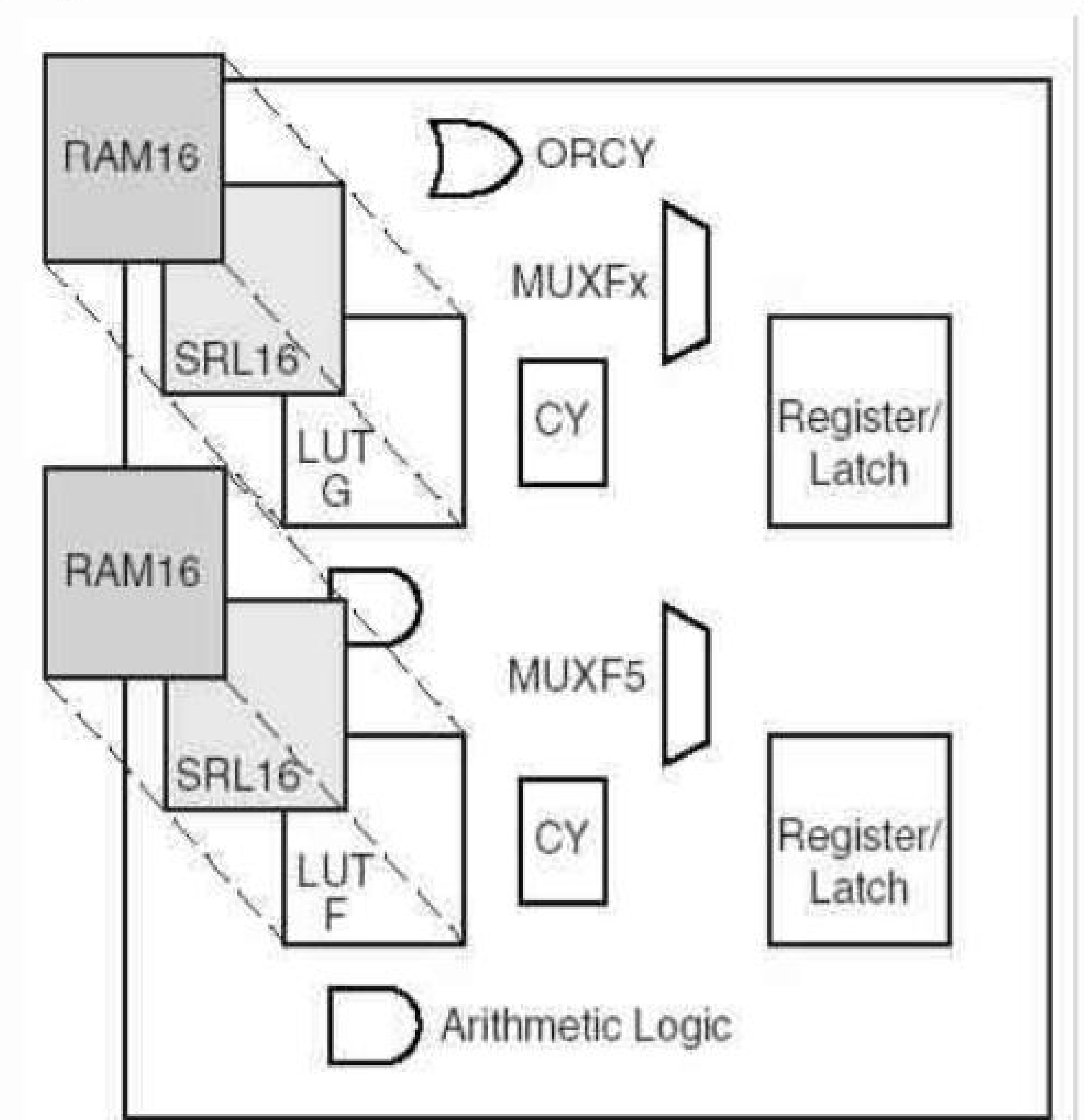
硬件协同仿真

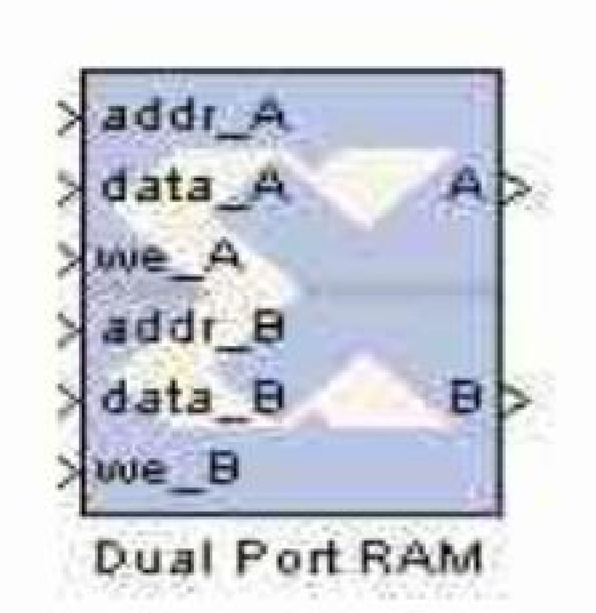

System Generator: mask
Xilinx System Generator
Compilation :
HDL Netlist
NGC Netlist
Bitstream
EDK Export Tool
Hardware Co-Simulation
Lyrtech
Timing Analysis
ML402
MicroBlaze Multimedia Board
XtremeDSP Development Kit
New Compilation Target...
Settings...
Part
XST
VHDL
FPGA Clock Period (ns) :
10
Clock Pin Location :
Create Testbench
Import as Configurable Subsystem
Provide clock enable clear pin
Override with Doubles :
According to Block Settings
Simulink System Period (sec) :
1
Block Icon Display:
Default
Generate
OK
Apply
Cancel
Help

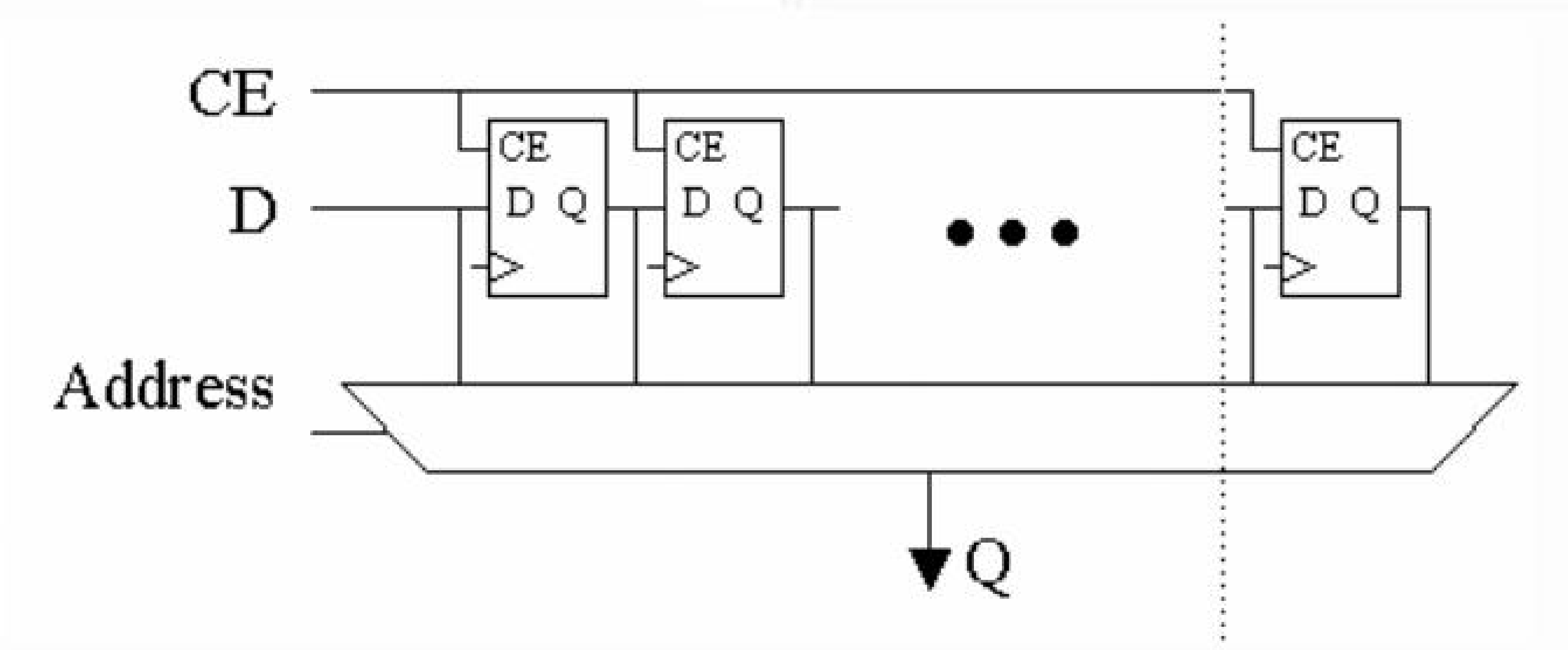


用 System Generator进行硬件设计

- FPGA 简介
- System Generator对FPGA的抽象
- System Generator对工程师的意义
  - System Generator的使用
  - System Generator和HDL的关系
  - HDL testbench
  - Co-simulation的意义



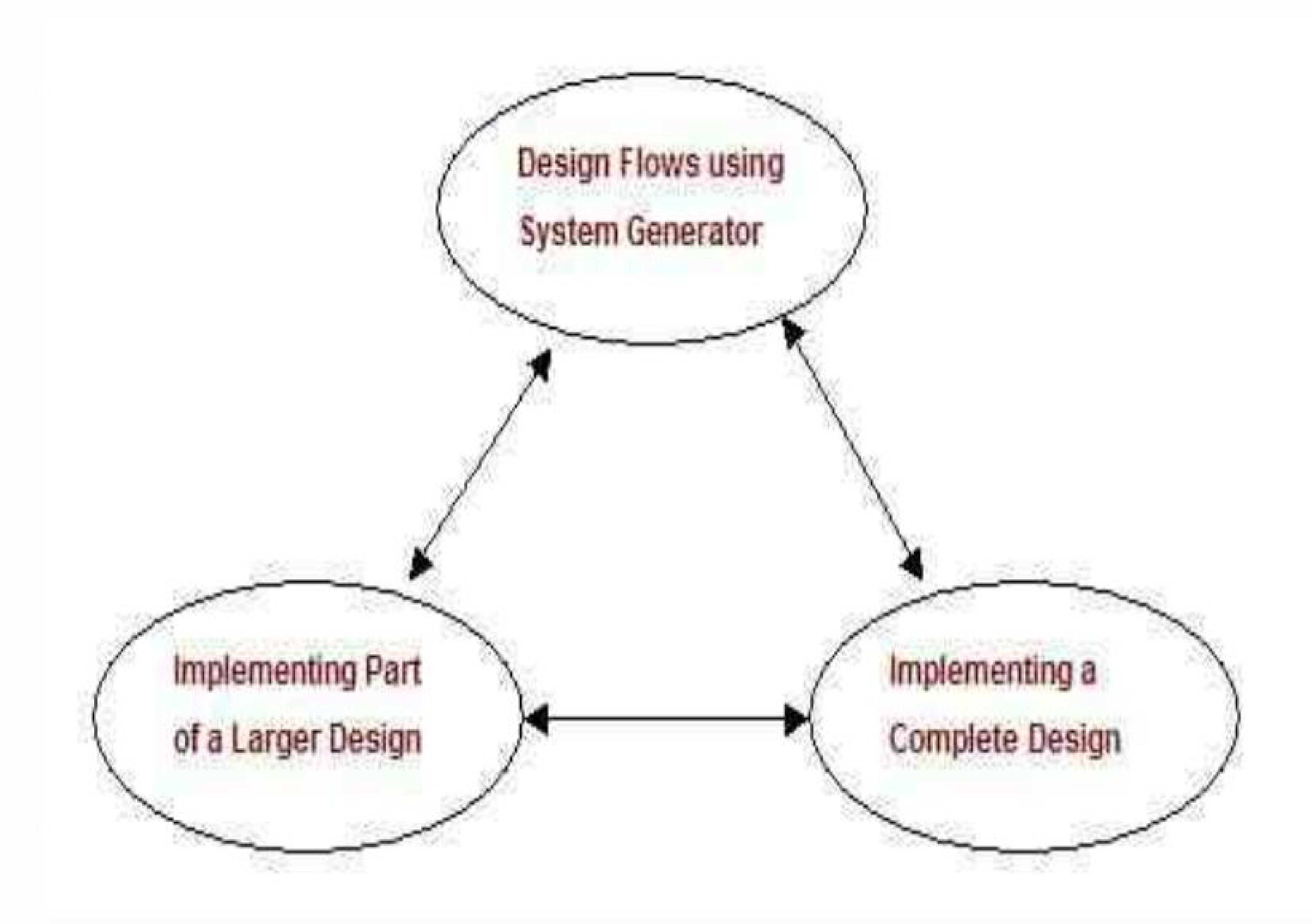






System Generator的设计流程

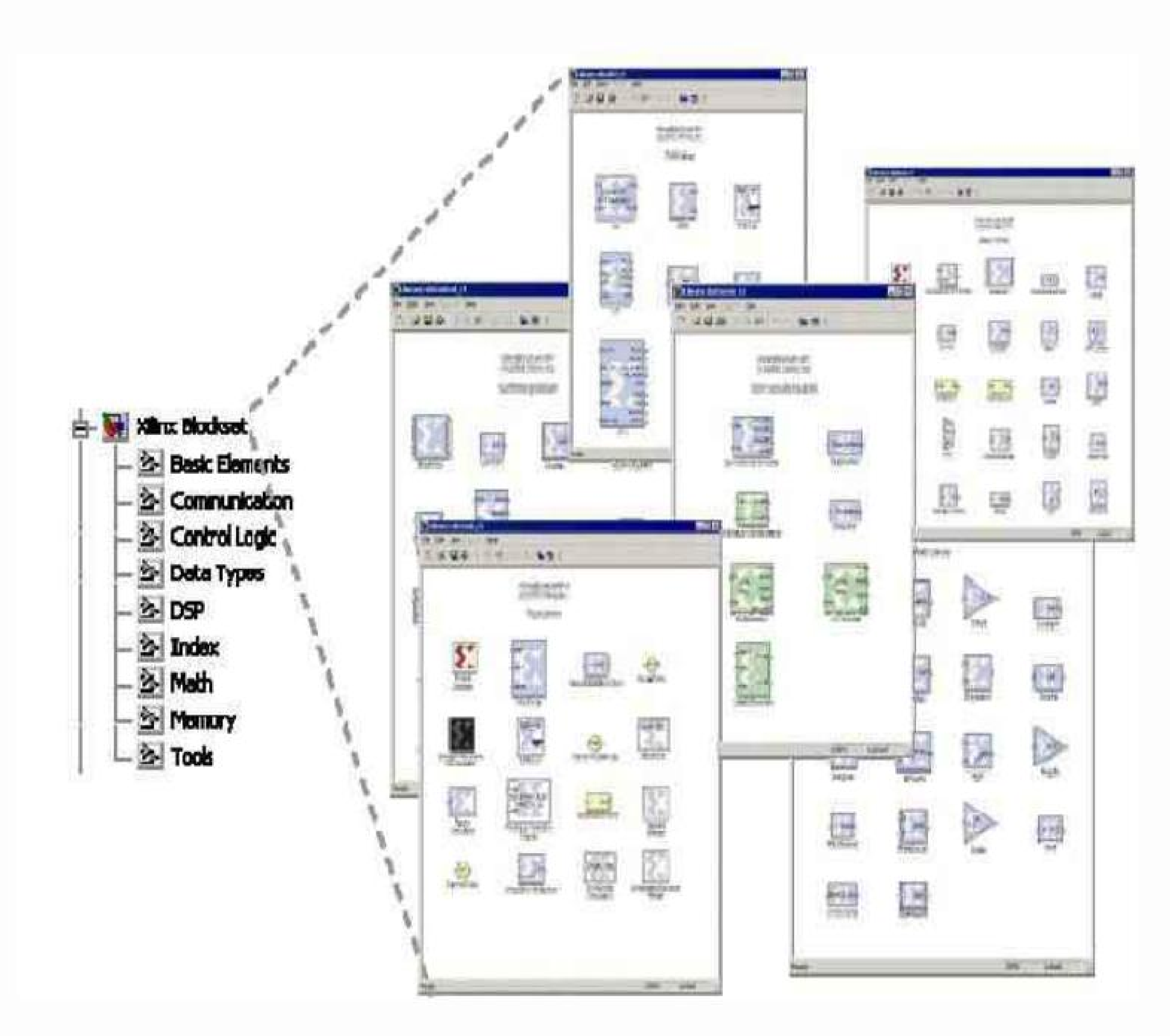
- 算法的开发
  - 实现算法的具体设计
  - 估计资源的消耗
- 大系统的部分实现
  - Co-simulation
  - HDL Wrapper
- 整个系统地实现
  - 实现设计的HDL
  - 时钟Wrapper
  - HDL testbench
  - Project文件
  - Project导航文件





System Generator 模块

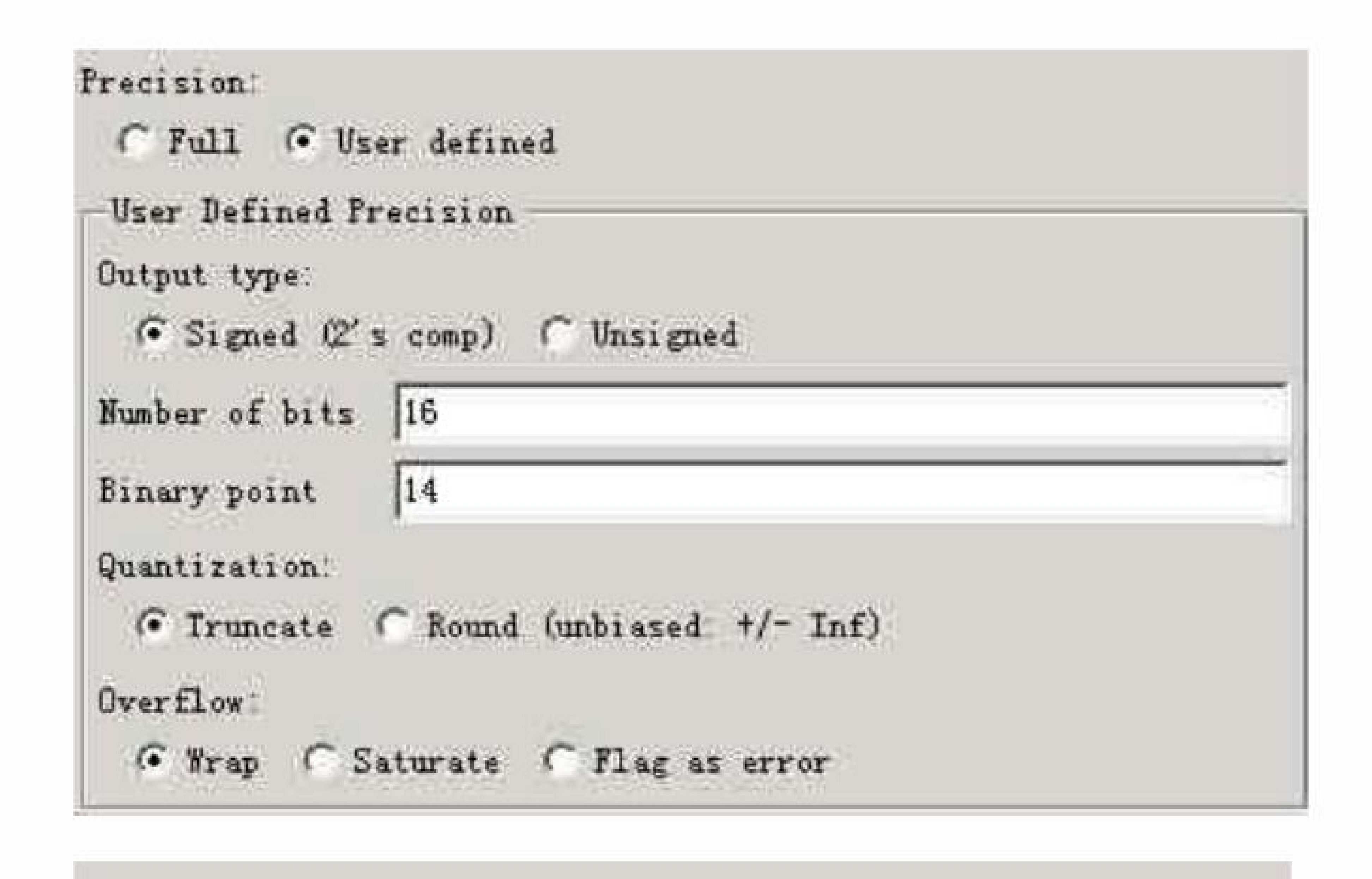
- ***Xilinx Blockset***
  - *Index*
  - *Basic Elements*
  - *Communication*
  - *Control Logic*
  - *Data Types*
  - *DSP*
  - *Math*
  - *Memory*
  - *Shared Memory*
  - *Tools*
- **Xilinx Reference Blockset**
  - *Communication*
  - *Control Logic*
  - *DSP*
  - *Imaging*
  - *Math*





模块的常用选项

- 信号类型
  - full precision
  - User-specified
  - Overflow options
- Sample Time
  - -1用于inherit
  - customization
  - Assert blocks
- Bit-True 和 Cycle-True 的概念

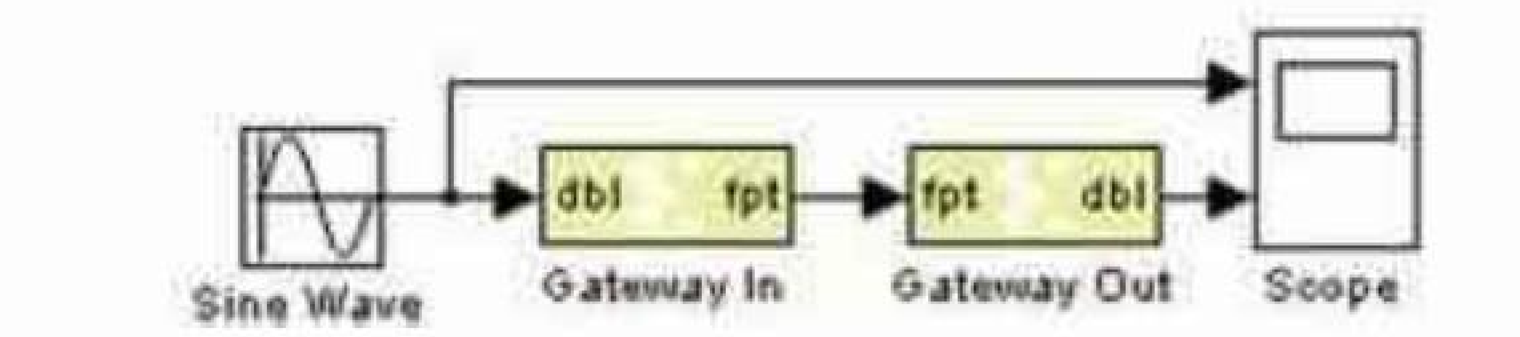


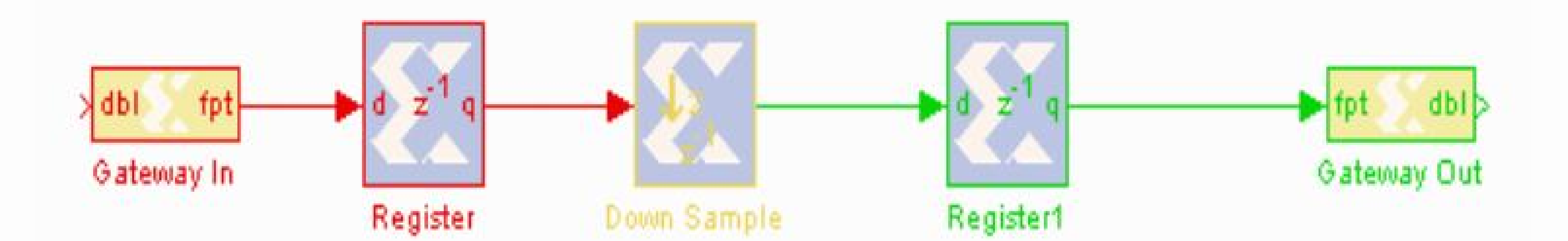


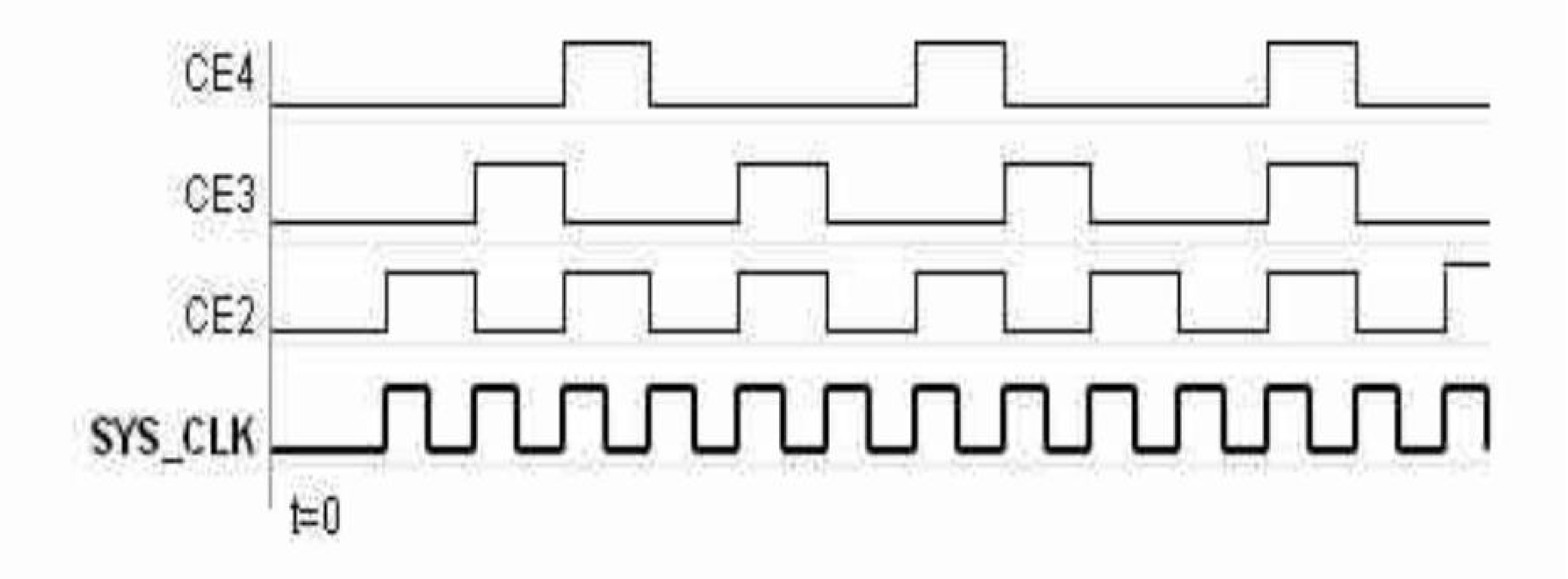


时序与时钟

- 离散时间系统
- 多速率模型
  - 变速率模块
  - 硬件过采样
- 异步时钟实现
  - FIFO
- 同步时钟
  - 时钟使能
  - Simulink时钟
  - FPGA时钟

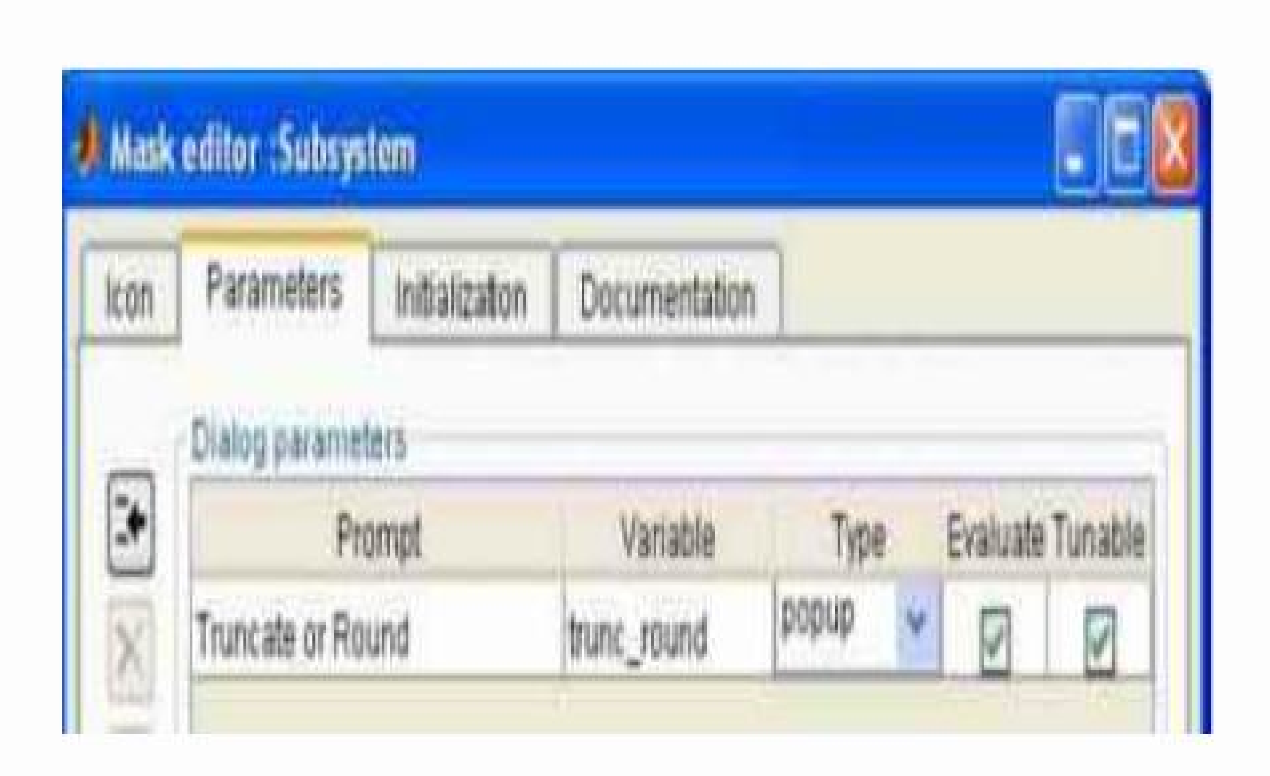




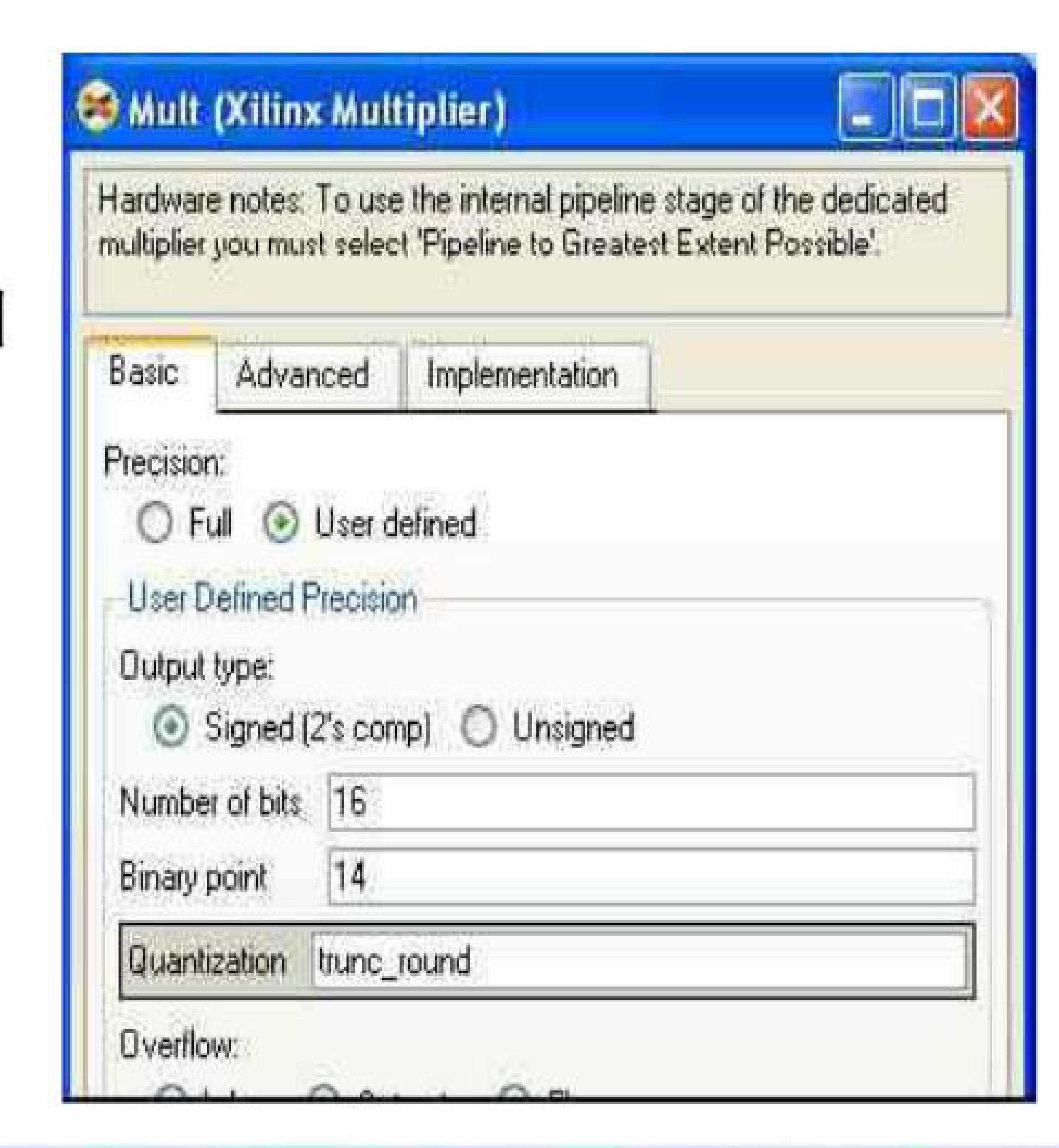




同步机制和传参机制



- 同步机制　Systemgen没有提供一个专门用于同步的机制,所以同步的任务主要是由设计者来完成的,但是systemgen也提供了一些用于同步的方法
  - Valid端口
  - 不确定值



- 传参机制　Systemgen的范围和参数传递方式和simulink一样,也就是说我们可以用变量和表达式来设置参数,这样我们可以利用matlab强大的数学功能来进行参数设置
  - 模块Mask
  - 参数传递



资源估计

- Systemgen提供了专门用于资源估计的工具,可以估计slice,lookup table,flip-flops,block memory,嵌入式multipliers, I/O block,三态buff等资源



Resource Estimator (Block Parameters)

Xilinx Resource Estimator

Slices	16
FFs	32
BRAMs	0
LUTs	0
IOBs	64
Emb. Mults	0
TBUFs	0

☑ Use area above

Estimate options: Estimate ▼ | Estimate

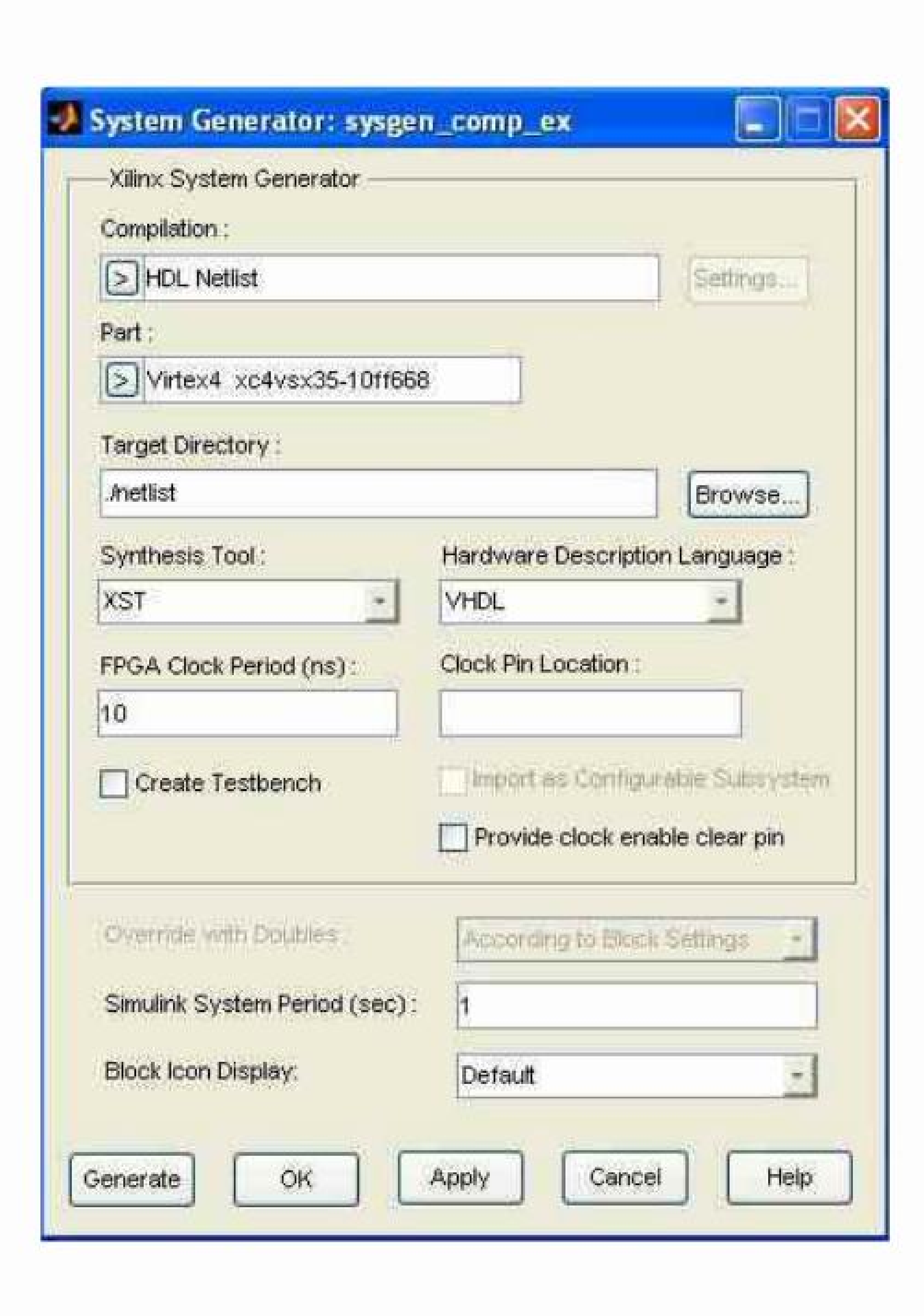
OK | Cancel | Help | Apply



自动代码生成

System Generator Block

- Compilation Type
  - HDL和NGC
  - 协同仿真
- Part
- Target Directory
- Synthesis Tool
- Hardware Description Language
- FPGA Clock Period
- Clock Pin Location
- Create Testbench
- Import as Configurable Subsystem
- Simulink System Period
- Block Icon Display
- Hierarchical Controls





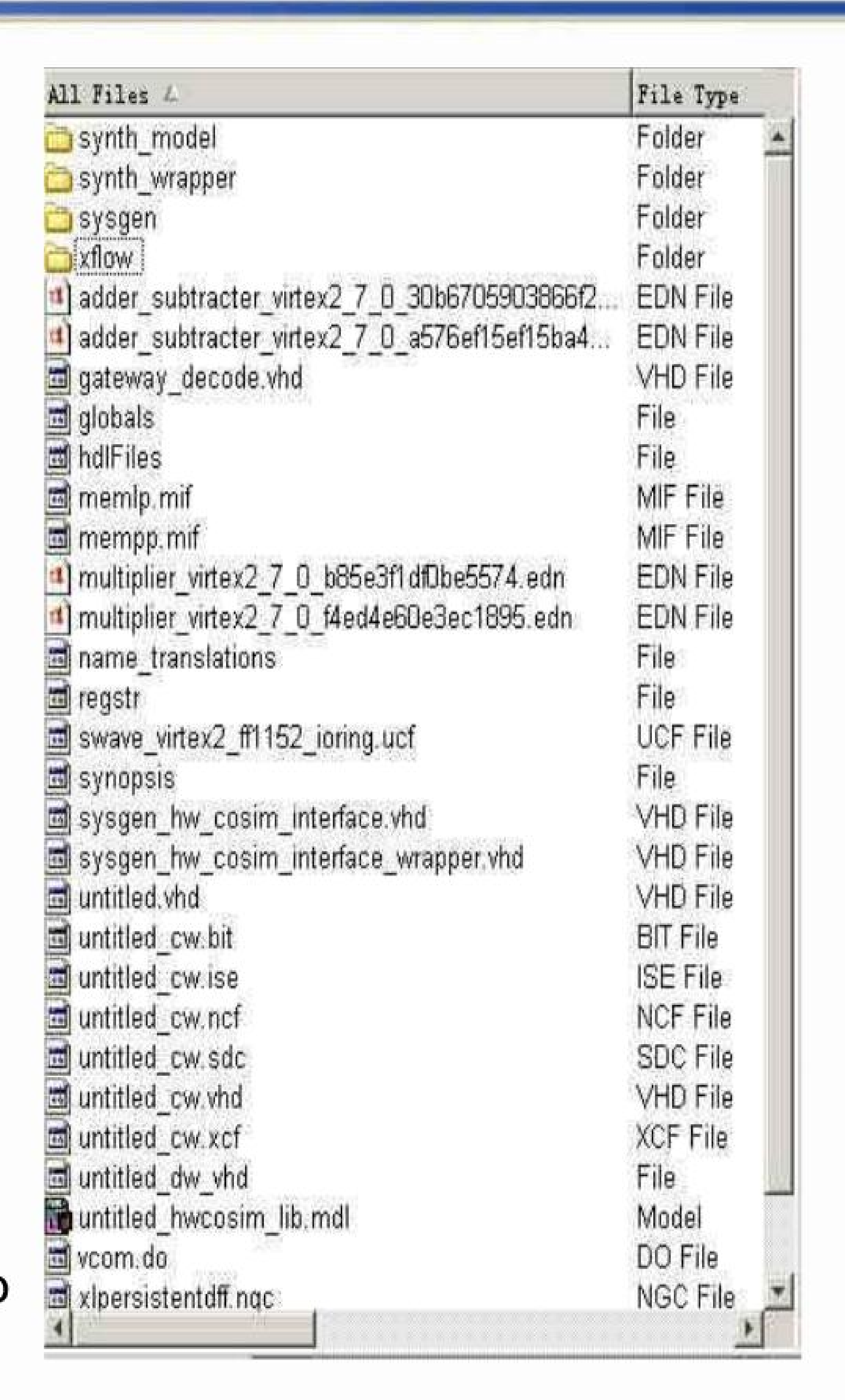
编译结果

不需要testbench下的主要文件

- <design>.vhd/.v
- <design>_cw.vhd/.v
- .edn and .ngc files
- globals
- <design>_cw.xcf (or .ncf)
- <design>_cw.ise
- hdlFiles
- synplify_<design>.prj / xst_<design>.prj
- vcom.do

需要testbench下的新增的主要文件

- <design>_tb.vhd/.v
- vsim.do
- pn_behavioral.do/pn_postmap.do /pn_postpar.do/pn_posttranslate.do





使用systemgen的约束文件

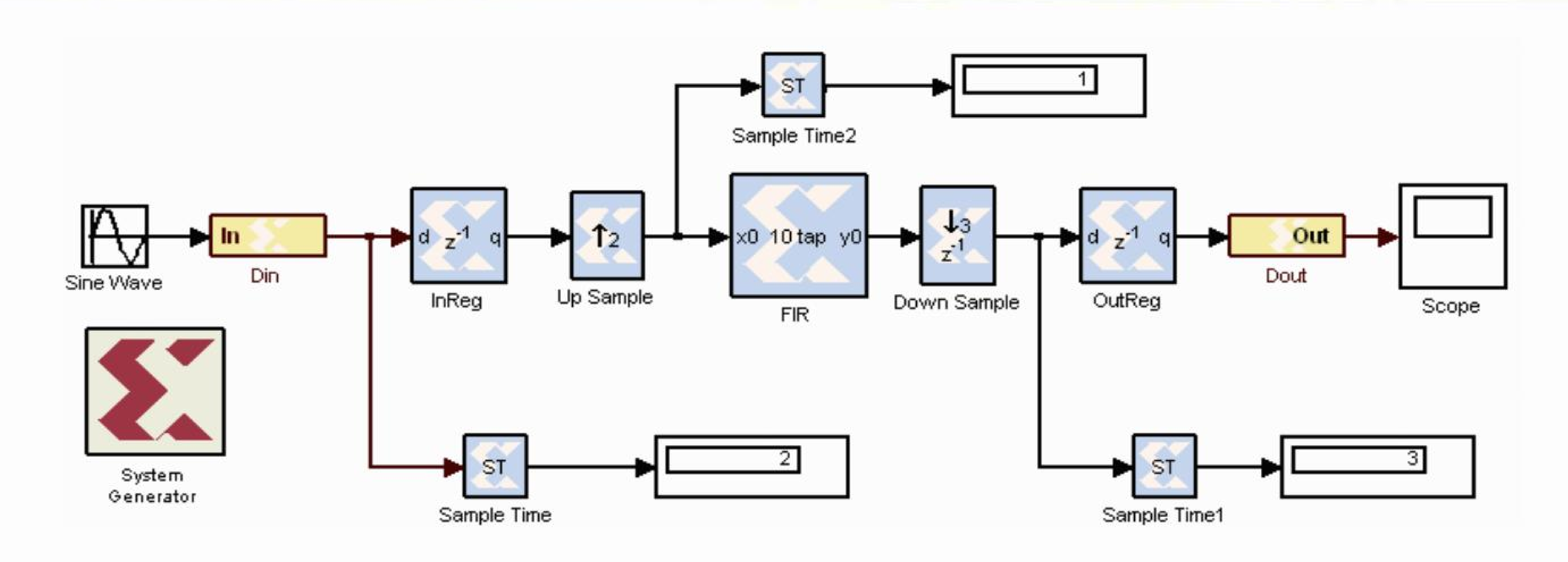
当设计完成后,sysgen会产生约束文件,告诉下载工具如何进行处理以保证高质量的处理,并使用更少的时间,这些约束主要包括:

- 系统时钟周期
- 各个子模块的运行速度
- 针脚的设置
- 各个端口的速度

约束举例

- System Clock Period
- Multicycle Path Constraints
- IOB Timing and Placement Constraints






```
# Global period constraint
NET "clk" TNM_NET = "clk_392b7670";
TIMESPEC "TS_clk_392b7670" = PERIOD "clk_392b7670" 10.0 ns HIGH 50 %;
# ce_2_392b7670_group and inner group constraint
Net "ce_2_sg_x0*" TNM_NET = "ce_2_392b7670_group";
TIMESPEC "TS_ce_2_392b7670_group_to_ce_2_392b7670_group" = FROM
"ce_2_392b7670_group" TO "ce_2_392b7670_group" 20.0 ns;
```

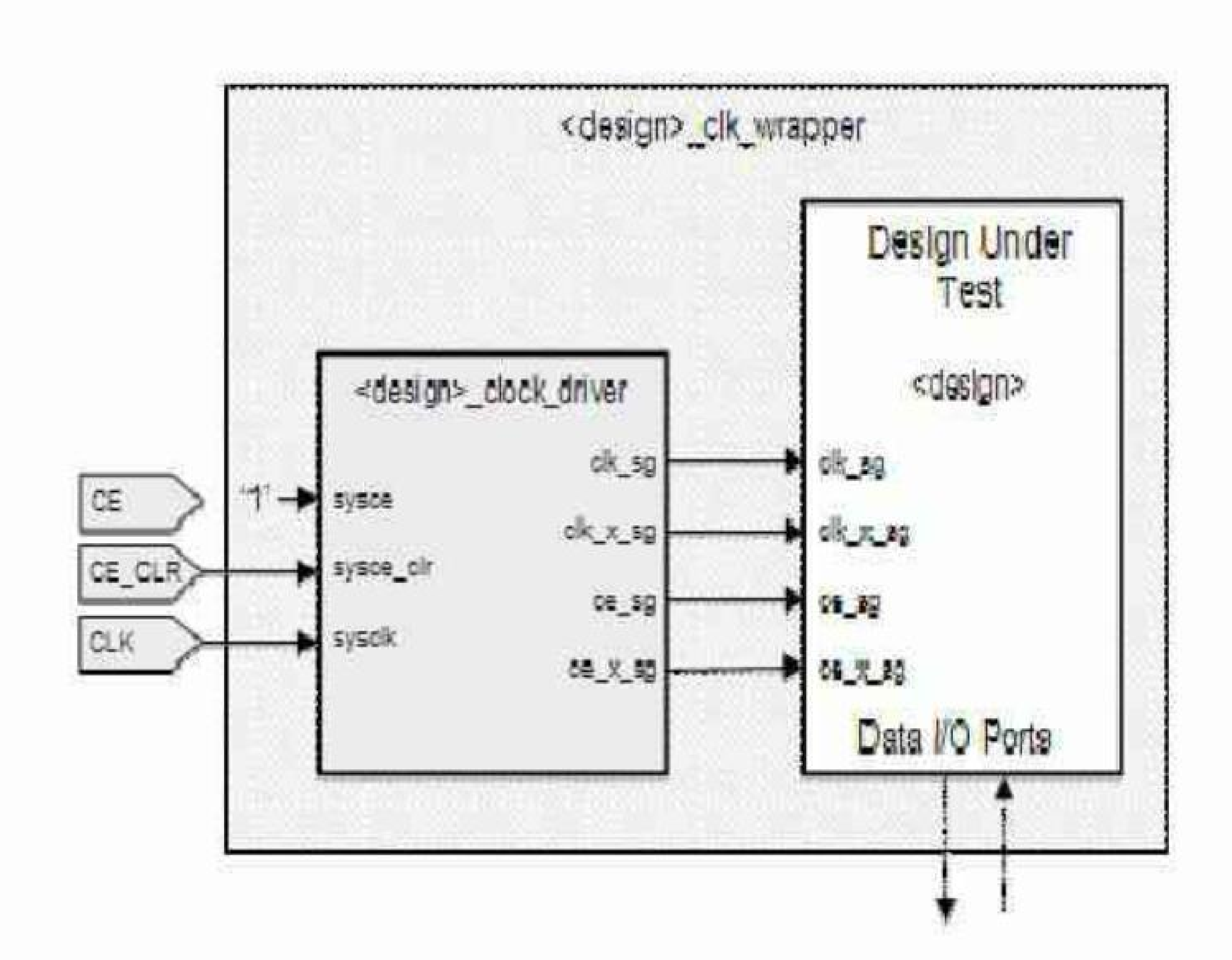



```
# ce_3_392b7670_group and inner group constraint
Net "ce_3_sg_x0*" TNM_NET = "ce_3_392b7670_group";
TIMESPEC "TS_ce_3_392b7670_group_to_ce_3_392b7670_group" = FROM
"ce_3_392b7670_group" TO "ce_3_392b7670_group" 30.0 ns;
# Group-to-group constraints
TIMESPEC "TS_ce_2_392b7670_group_to_ce_3_392b7670_group" = FROM
"ce_2_392b7670_group" TO "ce_3_392b7670_group" 20.0 ns;
TIMESPEC "TS_ce_3_392b7670_group_to_ce_2_392b7670_group" = FROM
"ce_3_392b7670_group" TO "ce_2_392b7670_group" 20.0 ns;
# Offset in constraints
NET "din(0)" OFFSET = IN : 20.0 : BEFORE "clk";
NET "din(0)" FAST;
NET "din(1)" OFFSET = IN : 20.0 : BEFORE "clk";
NET "din(1)" FAST;
NET "din(2)" OFFSET = IN : 20.0 : BEFORE "clk";
NET "din(2)" FAST;
…
# Loc constraints
NET "din(2)" LOC = "D35";
NET "din(1)" LOC = "B36";
NET "din(0)" LOC = "C35";
…
```



HDL中的时钟控制

- 时钟的命名
- 控制时钟生成的文件
  - <design>_cw.vhd/.v
- 合成更大设计时
  - 时钟用法
- CLK_(No.)和CE(No.)
- Clock wrapper的组成
  - 用于Design
  - Clock driver





Core Caching

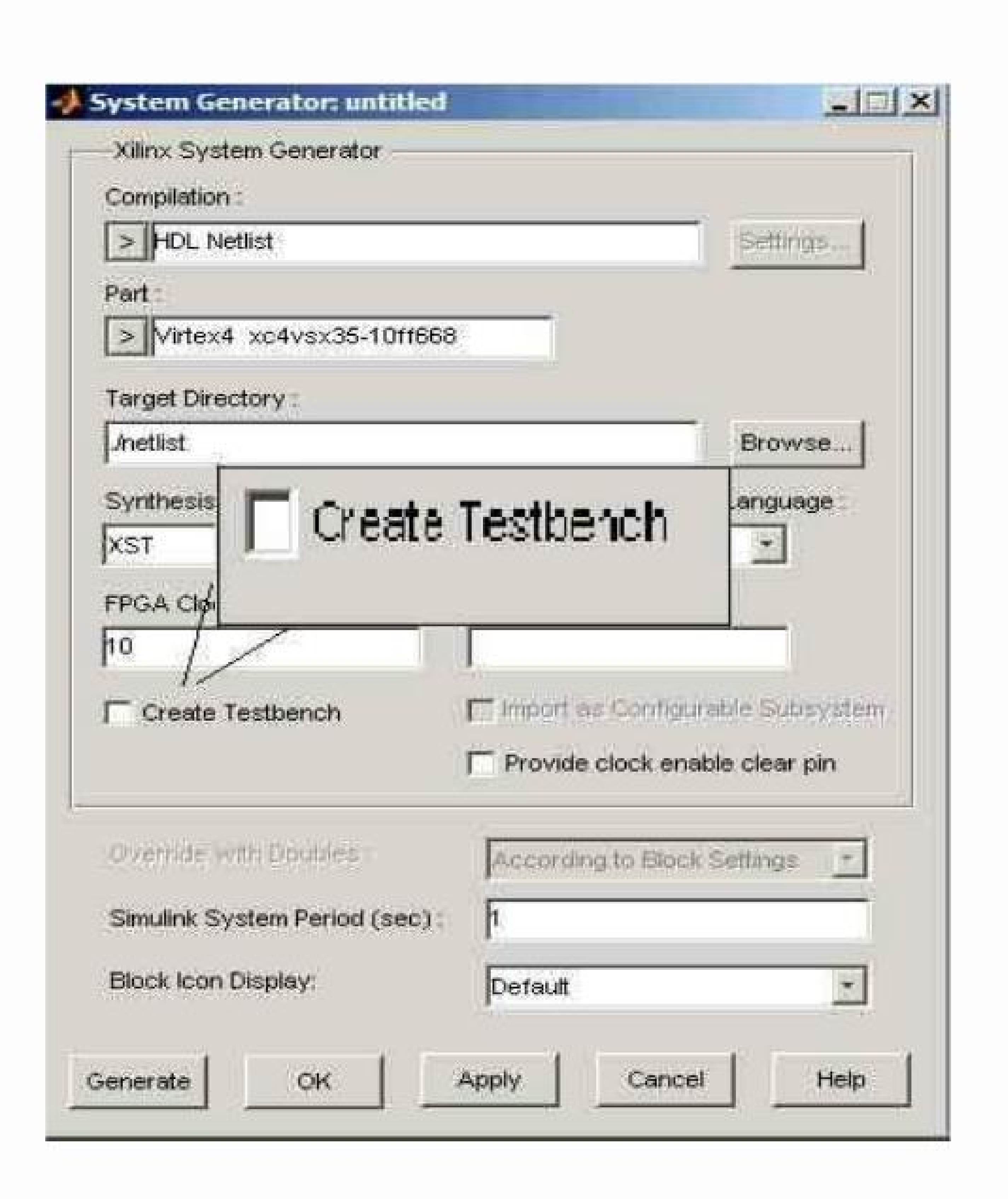
- Systemgen使用xilinx的CORE Generator (coregen) 来生成cores以实现部分设计
- Systemgen会预先生成一些cores，在调用coregen之前他会收缩这个cache,如何cores已经生成，那么就可以直接重用它
- Cache位于<sysgen>/xilinx/sysgen/core_cache，一般来说cache中的cores最多为2000个，达到限制后会自动删除多的
- 环境变量可以更改cache的位置和cache的大小限制，主要的变量包括：

Environment Variable	Description
SGCORECACHE	Location to store cached files. Setting this variable to a string of blanks instructs System Generator not to cache cores.
SGCORECACHELIMIT	Maximum number of cores to cache.



HDL Testbench

- Testbench用于将Simulink和HDL仿真器的结果进行比对，特别是当系统中有黑盒子的时候，生成的文件包括
  - 包含Testbench HDL实体的 <design>_tb.vhd/.v
  - 大量.dat文件,包括测试向量,激励,期望结果,以用于HDL Testbench仿真
  - VCOM.do和VSIM.do文件,可以在Modelsim中编译,并在Testbench中仿真





输入编译结果

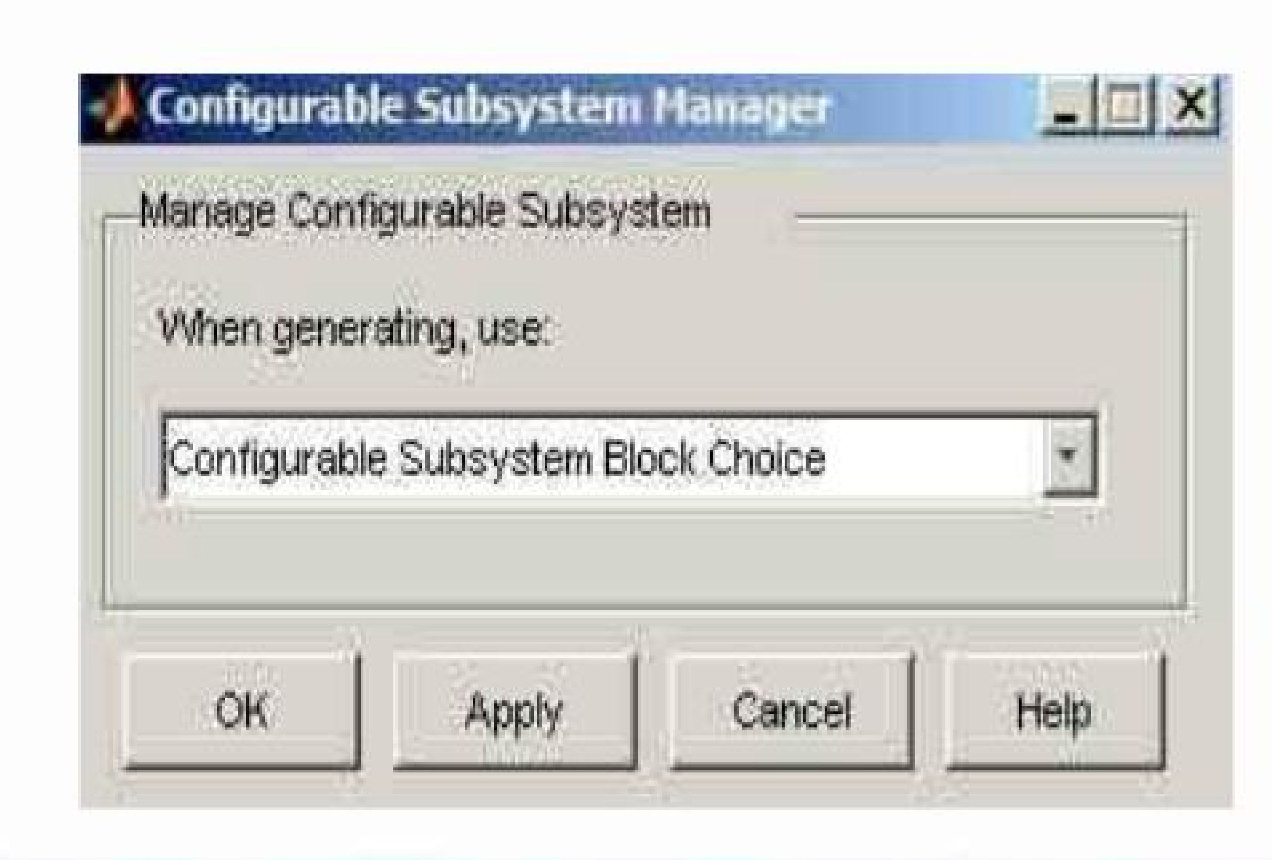
- 将编译结果输入设计可以指示System Generator完成下面的工作
  - 把子系统编译成HDL和EDIF
  - 使用黑盒子关联文件
  - 插入黑盒子和子系统
  - 用可配置子系统代替原始子系统
  - 告诉system generator 在后续的编译过程中,黑盒子中的文件必须使用
- 这个过程通常被称为incremental netlisting,它可以加快后续的编译工作,但是它也有些缺陷
  - 可能会使设计变得复杂
  - 使黑盒子失真不能真实地反映子系统,这种情况下要重新生成代码
  - VHDL生成的黑盒子不能在Verilog的incremental netlisting中使用，反之亦然



添加 Netlist

- 对子系统进行netlist incrementally,需要
  - 在子系统中拖拽一个System Generator block
  - 在其GUI中选择Compilation为HDL Netlist，打开Import as Configurable Subsystem勾选项
- System Generator把子系统翻译为HDL和EDIF，并关联到黑盒子上．然后可配置的子系统代替子系统（包含原始子系统和黑盒子）
  - 双击可配置子系统打开原始子系统
  - 打开configurable subsystem manager模块选择black box为生成选项







课程小结

- Simulink建模仿真
- 面向DSP的自动代码生成
- 面向FPGA的自动代码生成